Sparse and Low Rank Approximations for Action Recognition

Sushma Bomma

A thesis presented for the degree of
Doctor of Philosophy

Heriot-Watt University
Institute of Sensors, Signals and Systems
School of Engineering and Physical Sciences

July, 2016

Abstract

Action recognition is crucial area of research in computer vision with wide range of applications in surveillance, patient-monitoring systems, video indexing, Human-Computer Interaction and many more. These applications require automated action recognition. Robust classification methods are sought-after despite influential research in this field over past decade. The data resources have grown tremendously owing to the advances in the digital revolution which cannot be compared to the meagre resources in the past. The main limitation on a system when dealing with video data is the computational burden due to large dimensions and data redundancy. Sparse and low rank approximation methods have evolved recently which aim at *concise* and *meaningful* representation of data. This thesis explores the application of sparse and low rank approximation methods in the context of video data classification with the following contributions.

1. An approach for solving the problem of action and gesture classification is proposed within the sparse representation domain, effectively dealing with large feature dimensions,

2. Low rank matrix completion approach is proposed to jointly classify more than one action

3. Deep features are proposed for robust classification of multiple actions within matrix completion framework which can handle data deficiencies.

This thesis starts with the applicability of sparse representations based classification methods to the problem of action and gesture recognition. Random projection is used to reduce the dimensionality of the features. These are referred to as *compressed* features in this thesis. The dictionary formed with compressed features has proved to be efficient for the classification task achieving comparable results to the state of the art.

Next, this thesis addresses the more promising problem of simultaneous classification of multiple actions. This is treated as matrix completion problem under transduction setting. Matrix completion methods are considered as the generic extension to the sparse representation methods from compressed sensing point of view. The features and corresponding labels of the training and test data are concatenated and placed as columns of a matrix. The unknown test labels would be the missing entries in that matrix. This is solved using rank minimization techniques based on the assumption that the underlying complete matrix would be a low rank one. This approach has achieved results better than the state of

the art on datasets with varying complexities.

This thesis then extends the matrix completion framework for joint classification of actions to handle the missing features besides missing test labels. In this context, *deep* features from a convolutional neural network are proposed. A convolutional neural network is trained on the training data and features are extracted from train and test data from the trained network. The performance of the deep features has proved to be promising when compared to the state of the art hand-crafted features.

To my family

Acknowledgements

The best and baffling moments of my doctoral journey have been shared with many people. I take this opportunity to thank everyone who have been with me and helped me directly or indirectly.

My first debt of gratitude goes to my supervisor Prof. Neil M. Robertson for his continuous support and encouragement. His valuable insights and suggestions at all times have helped a lot in carrying out this work effectively.

My sincere thanks to Prof. Paolo Favaro for taking me on as a Ph.D. student and for his invaluable technical support.

They both have provided the vision and advice necessary to proceed through this program and complete my dissertation. Their vast experience in the area of computer vision and machine learning, and their willingness to impart their knowledge has served me well. I owe them my heartfelt appreciation.

I honestly thank my examiners Dr. Toby Breckon, Reader in School of Engineering and Computing Sciences, Durham University, UK and Dr.Alexander Belyeav, Associate Professor in School of Engineering and Physical Sciences, Heriot-Watt University,UK, for reviewing this thesis and valuable suggestions made.

I gratefully acknowledge the funding sources, Heriot Watt University for my Ph.D. program, School of EPS, BMVA (British Machine Vision Association) and IEEE for my travel grants to attend the conferences.

A big thanks goes to Al, Reza, IT department and Lynn for providing the resources and help whenever needed.

It has been a great privilege to be a part of Vision Lab, Heriot-Watt University and all its past and present members will always remain dear to me. Their friendship and assistance has meant more than I could ever express.

My sincere thanks goes to Rick for his help in implementing some codes, Rolf and Deepayan for their suggestions in preparing for conferences.

Special thanks goes to Eleonora, her invaluable assistance and guidance right from the day one in the lab mean a lot. I could not have completed my work without her help.

Furthermore, thanks to Roushanak for cheering up in my tough days.

I am always indebted to all my friends in Edinburgh, who have always offered help at right times by taking care of my kids whenever asked. Specially Soni for her help within the university and outside.

Special thanks to my grandmother, aunts and uncles in India for their encouragement and advice throughout and cousins who have cheered up with their messages and conversations and with whom I had good time.

My heartfelt thanks to my dear mum and mother-in-law who came to my place all the way from India to look after me and my family in difficult times. I am always grateful to my brother and his family for all the help he has provided in this regard and more.

Finally, my family. I thank God for blessing me with a wonderful family. They are with me all the time. My two daughters have been a source of happiness and light throughout. Just a simple thanks would never be sufficient to my beloved husband, Syama. I would not have had the precious opportunity of taking this exciting journey without him. It is his love, support, motivation and patience which has made the rocky road smooth.

Sushma Bomma

Contents

List of Tables

List of Figures

List of Acronyms

2D Two dimensional . . . 19

3D Three dimensional . . . 12

4D Four dimensional . . . 19

AI Artificial Intelligence . . . 75

BoW Bag of Words . . . 5

BoVW Bag of Visual Words . . . 20

BPDN Basis Pursuit De-noising . . . 29

CNN Convolutional Neural Network . . . IX

CS Compressive Sampling or Compressed Sensing . . . 3

DBM Deep Boltzmann Machine . . . XI

DBN Deep Belief Net . . . XI

DS Downsampling . . . IX

EM Expectation Maximization . . . 30

FFT Fast Fourier Transform . . . 12

FPC Fixed Point Continuation . . . 59

FV Fisher vector . . . 21

GAs Greedy Algorithms . . . 25

GEI Gait Energy Image . . . 40

GMM Gaussian mixture model . . . 21

HCI Human Computer Interaction . . . 1

HOF Histogram of Optical Flow . . . 16

HOG Histogram of Oriented Gradients . . . 15

HWU Heriot-Watt University . . . IX

JL Johnson-Lindenstrauss . . . 42

Chapter 1

Introduction

Vision is the dominant sense of perception in human beings. Identifying an image is as fast as 13 milliseconds according to recent studies at MIT (Source : http://news.mit.edu/), considering the fact that the content in the images is previously learnt. The astonishing speed with which human brain can process inputs creates a delusion that it is multi-tasking, when actually it cannot [4]. While recognizing an unknown or unseen event for a human itself is challenging without prior knowledge, it is daunting task to realize artificial systems which can mimic human vision. Vision based research on human action recognition dates back to 1970s and has been very active owing to its numerous applications including surveillance, gaming, Human Computer Interaction (HCI), video-indexing. Among which video surveillance and console gaming occupy major share.

Video surveillance industry has been growing rapidly as a result of increasing crime rate. According to surveillance industry statistics, it is expected to increase by 12% from 2013 to 2018 reaching $23.6 billion revenue worldwide (Source: https://technology.ihs.com/. Surveillance cameras these days are not only needed in public places like airports, hospitals, shopping malls etc, but they are also being used for private purposes like factory surveillance, elder care and so on. Figure 1.1 shows images from recordings in different surveillance scenarios.

A typical surveillance system consists of one or more camera arrays which capture videos in multiple views and these are simultaneously observed on number of screens at a surveillance station as shown in Figure 1.3 . Most of these systems require monitoring round the clock to capture any criminal or abnormal activities requiring a human operator. The data generated from such cameras is enormous and often need to be archived for future, creating severe bottlenecks in storage, analysis and network performance. Coming to gaming applications, although mo-

Figure 1.1: Surveillance camera footages in various environments.
From left showing sample images from recordings at airport parking, elder care home, factory and hospital. Images source - Public Domain

Figure 1.2: Console gaming
From left showing a Microsoft Xbox. Center and right show console gaming being used for physical therapy of the patients in a medical center and right shows the entertainment aspect of gaming. Images source - Public Domain

bile devices are trendy recently, Personal Computer (PC) based gaming continue to be still in demand making an 8% increase in the global market in 2015 and TV, console and VR gaming to increase by 2% according to the games industry report (Source: www.gamesindustry.biz/). Console gaming is widely used in health care and fitness centres also besides being a recreation tool. Example consoles include Microsoft Xbox, Nintendo Wii, Sony playstations etc., These are connected to a PC or TV monitor and are provided with a joystick and a remote control. The consoles have inbuilt cameras which capture the action and responds to it instantaneously following on some game rules. The recognition must be robust to motion variation, clothing, lighting and camera angle.

These applications and more, relying on action recognition from videos, require handling large amounts of data. Before actually analysing the data for a given application, it would be sensible to know whether all the data that has been recorded or being recorded contains relevant information. For example if we observe the recordings from a surveillance video as in Figure 1.3, there is a lot of redundancy. Similarly with the gaming applications, the camera picks up the complete view in which the required action is just some percentage of the total view. Suppose by some means the redundant data has been removed. The next task is to analyse

Figure 1.3: Typical surveillance system components
From left showing camera array, recordings from such cameras and an operator monitoring the videos at the surveillance station. Surveillance requires continuous monitoring of these recordings by a human operator. Images source - Public Domain

the data to identify a specific action. Practically one cannot completely avoid data redundancy. Moreover, in real scenarios, data is often corrupted or incomplete. Within this situation, one can raise a question that if the redundancy cannot be eliminated could this be exploited in some way to effectively analyse the data? If so how and to what extent in real situations. Surveillance systems currently require an operator to analyse and identify multiple actions simultaneously. Will such exploitation help to simultaneously recognize multiple actions which is actually not possible even for the human brain? This thesis aims to provide solutions to these questions.

The goal of this thesis is to design a robust action recognition system towards identifying multiple actions simultaneously exploiting the inherent redundancy in video data .

1.1 Motivation

Despite great progress over past years, the gap between human perception and *computer's vision* is still wide in this field, giving scope for new methods. Data compression methods like Joint Photographic Experts Group (JPEG), Compressive Sampling or Compressed Sensing (CS) [5] have become popular lately owing to the developments in digital networks and sensors which generate plethora of data. The present big data era seeks methods which can store, process and transmit data efficiently. CS alleviates this by looking for meaningful structures in data residing in much lower dimensions. Most natural images or signals have a low dimensional structure or exhibit sparsity in specific bases like Fourier, Wavelet etc. CS relies on the *sparsity* of the data and was introduced to vision community around 2006 [5]. It is the process of acquiring data by sampling very few measurements from domain incoherent to the sparsifying domain. CS has been primarily

Figure 1.4: User rating data
Movie rental company Netflix has announced a challenge to come up with a recommender system based on the highly incomplete user-rating data. Images source - Public Domain

developed for the robust signal reconstruction where the acquisition is expensive or difficult. Magnetic Resonant Imaging (MRI) is the predominant application based on CS [6]. Other applications include Network Tomography [7], Infrared Cameras, Single pixel imaging [8]. An upshot of CS, which has gained popularity recently is *Matrix completion*. It is the task of predicting the unknown or unobserved entries in a data matrix based on the *dependencies* within the data. Matrix completion is popularly associated with recommender systems. These are systems which recommend products to users by predicting the users' choice depending upon the ratings or reviews on that product and the users' history. Popular examples include Netflix [9] - movie rental company, Amazon - online shopping website. The design of recommender systems is solely based on the highly incomplete user rating data. Figure 1.4 shows a pictorial representation of user ratings for different movies . The strong assumption in matrix completion is that the underlying matrix is of low rank. Sparse representations of the data and matrix completion methods have been adopted by the computer vision community for applications such as object detection [10], face recognition [11], image classification [12] and the results have been encouraging. Motivated by this, we look for solutions to the above mentioned data redundancy problem in action recognition applications using the two main paradigms of CS, sparse and low rank approximations.

In this work, our goal is to explore the applicability of compressed sensing tools to extract the underlying sparse and low rank structures in video data towards designing a robust action recognition system.

1.2 Challenges and Objectives

The technical challenges posed towards building an action recognition system are discussed now. Figure 1.5 shows the pipeline of a typical action recognition system. Each block in the figure is a wide research area on its own. For a given application, building such system starts with collection of ample relevant video data. The data is split into training, validation and test data. The performance of the system primarily relies on four areas - detection, tracking, feature extraction and classification. Detection is identifying where in the entire video an action of interest has occurred for the first time and tracking is to follow that action throughout the video. These two elements are not shown in the figure assuming that the training data available is the output from a tracker. Each training video is then represented as a set of features which capture the video content (*action*) in the best possible way. To reduce the computational burden, the set of features are encoded, such that the dimensionality is reduced without a great deal of information loss. A classifier is trained to learn the actions in the training data from the encoded features. The parameters of the classifier can be tuned with the validation data. After training the performance of the classifier is evaluated with test data which the classifier has not *seen* before.

The typical challenges posed in action recognition are (1)View-point variations due to moving cameras [13] (2)Dynamic environments like changing or moving backgrounds and human and non-human occlusions (3)Anthropometric differences and various styles of executing an action by humans (4)Insufficient training data (5)Different levels of actions. [14]

The dominant challenges in realising a real time action recognition system are high dimensionality and redundancy of data. Various dimensionality reduction methods like Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Fisher encoding, Bag of Words (BoW) etc., have been proposed in the literature to overcome the dimensionality issue [15] [16]. However, the main drawback of these methods is that they are data-dependent and hence the encodings need to be determined every time the training data is modified. Another concern which has not been addressed well in this field is joint or simultaneous classification of more than one action [14]. Many applications including surveillance, HCI environments, gaming and patient-monitoring systems would benefit with simultaneous classification of actions where the entire video may not contain relevant information and could be of low resolution. Towards realising the proposed goal, the work in this thesis addresses these two areas in action recognition.

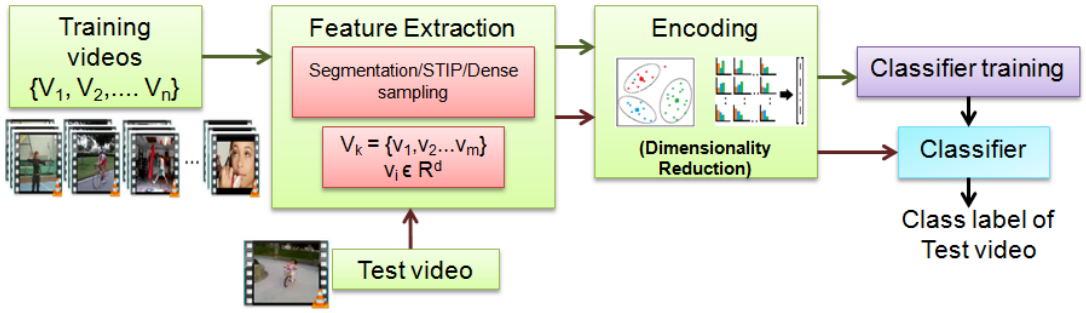

Figure 1.5: Pipeline of an action recognition system

Designing an action recognition system starts with collecting video data $\{V_1, V_2, \ldots V_n\}$. These videos are first converted into suitable representations by extracting features. The high dimensional set of features are encoded to a lower dimension. Finally a classifier is trained on the encoded features of the training data only. A test video is processed through same stages except the classifier training stage and its identity would be obtained from the previously trained classifier.

Specifically our objectives are

- to deal with the high dimensionality of features using a data independent dimension reduction method which can be generated efficiently
- to design a framework for classifying actions from multiple videos simultaneously even under severely constrained data situations.

1.3 Approach

The success of CS techniques lies in exploiting the underlying low dimensional structure of the data. Visual raw data is high dimensional (number of pixels in images or video), nevertheless the intended features for an application lie on low dimensional subspaces. Extracting such structures from video data is a challenging task. Our approach in this thesis explores applicability of sparse and low rank approximation methods addressing the specified objectives towards realising a robust action recognition system. Technically our approach is two-fold and can be stated as:

- *To exploit sparsity of features addressing the feature dimensionality problem from the principles of compressed sensing and classify actions based on sparse representations.*
- *Extend the notion of sparsity to low rankness of the data in the feature space to classify multiple actions simultaneously using matrix completion methods.*

We implement these methods on videos of generic actions with a single action performed by a single person.

1.4 Contributions

The following contributions are made towards the proposed goal based on the above mentioned approaches.

- A supervised classification method based on sparse representations of *compressed* features is proposed which effectively handles the large dimensional features. The approach performs better than the state of the art [17] [18] [19].
- Low rank matrix completion based classification is proposed to classify multiple actions from different videos jointly, within a transduction and supervised learning. Currently this framework has only been applied for multi-label image classification. [20].
- Deep features from a convolutional neural network are introduced. In the matrix completion setting, these features have shown to be more promising than the state-of-the-art hand crafted features. Matrix completion approach is extended to handle feature deficiencies using deep features and has shown consistent performance even with more than 70% missing features. This is comparable to the recent works [21] [22].

1.4.1 Publications

The main results of this work are published as the following.

- S. Bomma, P. Favaro, and N. M. Robertson, "Sparse representation based action and gesture recognition", *in Image Processing (ICIP), 2013 20th IEEE International Conference on, 2013, pp 141-145.*
- S. Bomma and N.M. Robertson, "Joint classification of actions with matrix completion", *in Image Processing (ICIP), 2015 IEEE International Conference on, Sept 2015, pp 2766-2770.*
- S. Bomma and N.M. Robertson, "Deep Action Classification via Matrix Completion", *24th European Signal Processing Conference - EUSIPCO 2016.*

1.5 Thesis Roadmap

The words *signal*, *data*, *sample*, *image*, *video* are used interchangeably corresponding to the context in this thesis. This thesis comprises of 6 chapters with 3 chapters of our contribution. It is structured as follows.

- **Chapter 2** presents the literature review on action recognition and sparse representation highlighting the recent trends and state-of-the-art in both domains individually. Then focus on various approaches on how sparse representations have been applied to classification problems in computer vision.

- **Chapter 3** presents our first part of our goal of applying sparse representations to the problem of action recognition, focussing on random projections to handle large dimensions of the features. We extract shape-based features and optical-flow based features from the video data. The proposed approach is evaluated on standard action dataset and gesture dataset released from our lab [23].

- **Chapter 4** introduces matrix completion framework. The analogies between sparse representation and matrix completion are highlighted. A joint classification method for actions is proposed using matrix completion techniques. The approach is evaluated on three datasets of varying complexity [24].

- **Chapter 5** proposes deep features from a convolutional neural network. Matrix completion framework is extended to deal with missing features scenarios using deep features. The performance of these features is found to be stable when compared to that of state-of-the-art hand crafted features in extreme situations of feature deficiency.

- **Chapter 6** summarizes our work and discusses some future aspects in the same direction.

The terms *recognition* and *classification* used in the thesis imply classification only. The performance or accuracy is measured in terms of overall recognition rate in all the settings.

Chapter 2

Background

This chapter reviews the theoretical background in action recognition and sparse representations. The details pertaining to the different blocks in an action recognition system shown in Chapter 1 of this thesis, Figure 1.5, are dealt first. Next, we explain the theory behind sparse representations and review their applications in various classification tasks. We conclude summarizing main points from both domains.

2.1 Action Recognition

Action recognition is the problem of assigning a label to an action class observed in a video. Extensive research has been focussed in this area owing to the large number of practical applications. An excellent review on the action research in the past decade can be found in [14, 25]. Most of this section is adapted from these works. An action can be defined as the accumulation of human body movements in some structured manner. Depending on the complexity of the structure, actions can be categorised as (1) Gestures - performed by single actor with minimal body movements (only a part of body is involved) extending for very short time duration, mostly used for automation control. Eg. 'start','stop'etc., (2) Actions - moderate body movement performed by single actor extending for more time duration than gestures. Eg. 'walking', 'waving'etc. (3) Interactions - either between two persons or between person and object. Eg. sports actions, punching etc. (4) Group activities - involving a group of persons. Eg. 'marching', 'protest', 'group stealing'etc. Several approaches have been proposed in this area. For any approach, first a video has to be represented into a suitable form (set of features). Appearance and motion are important cues in recognizing actions. Features which capture these cues in various ways along with the classification or recognition method followed are detailed below. The following subsections review some of the milestones in the literature.

2.1.1 Shape or appearance based global features

These features are computed taking into account complete human body. Silhouette based Motion Energy Images (MEIs) and Motion History Images (MHIs) were proposed in [29] to recognize human motion using temporal templates. MEIs are cumualtive binary images obtained by taking the difference between a initial frame and following key frames. MEIs are indicative of where in the video the motion is taking place. MHIs are a scalar valued images obtained by checking the temporal motion at each pixel. The pixel intensity value is higher (brighter pixel) if there has been a recent motion at that pixel. Both MEIs and MHIs are used as a temporal template for matching two motions. The reason for combining both as temporal template is that for some motions MEIs may be similar and for some MHIs may be similar as shown in Figure 2.2. Combining both discriminates motions efficiently. Training data is composed for movements and variety of viewing angles of each movement. An MEI and MHI pair is computed for each view and movement. A statistical model of the Hu moments is computed for each pair. A

Figure 2.1: Different levels of actions

First row : Snapshot of gestures from [26], Second row : Snapshot of simple actions performed by a single human [27], Third row: Sports actions with human and an object interaction from [28], Fourth row: Typical group activities. Images source - Public domain)

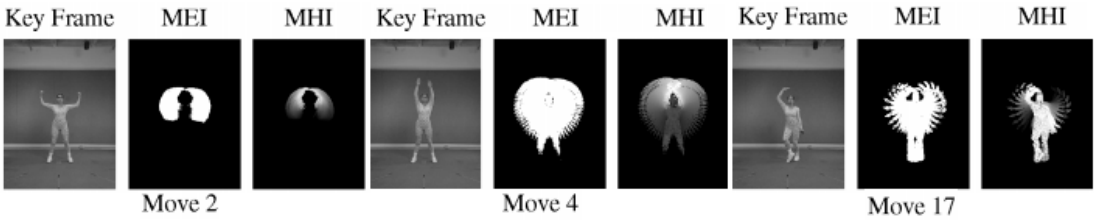

Figure 2.2: Motion Energy Images and Motion History Images

Key frames and corresponding MEIs and MHIs of Moves 2, 4 and 17 of a ballet dataset are shown. It is seen that Moves 4 and 17 have similar MEIs and Moves 2 and 4 have similar MHIs . Figure adapted from [29]

test input is recognized based on the *Mahalanobis* distance between the moment descriptions of the input to that of each training sample.

Actions are regarded as space-time shapes in [30] which contain spatial information of the pose of the human and location and orientation of the torso and different limbs. These shapes encapsulate the dynamic information of the global body motion and relative motion of the limbs. Local and global features are obtained by solving Poisson's equation applied to these shapes. The eigenvectors and eigenvalues from the Hessian of the solution relate to the local principal direc-

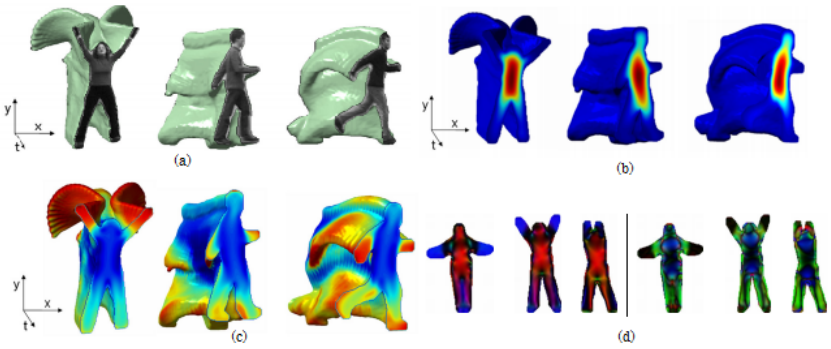

Figure 2.3: Action representation as space-time shapes
(a) Space-Time shapes of some actions (b) Solution of the Poissons equation on the shapes (c) Space-Time saliency (d) Space-Time orientations - Degree of 'Plateness'-Degree of 'Stickness'adapted from [30]

tion (expressed in terms of degree of *stickness* or *plateness*) and local curvature (expressed in terms of *ballness*) respectively. Global features are extracted by computing the weighted moments using a weighting function $w(x, y, t)$ and the characteristic function $g(x, y, t)$ of the space-time shape. Figure 2.3 illustrates the space-time shapes and features. Classification is done using nearest neighbour approach with leave one out cross validation. Euclidean distance measure between test and each of the training sequence is used the distance metric for classification.

In [31], conventional 2D Maximum Average Correlation Height (MACH) filters are generalised to be used with 3D space-time volumes. An action class is represented by a synthesized filter which fits the volume of the training data. For each video spatio-temporal frames which have single cycle of an action are considered as a volume and for each pixel temporal derivative is applied. Each volume is represented in the frequency domain by applying Three dimensional (3D) Fast Fourier Transform (FFT) at each pixel. After obtaining the FFT of each volume as a 3D matrix, each 3D matrix from an action class are vectorised into columns, resulting in as many columns as the number of training examples in each class. Next, an Action MACH filter h for that action class is synthesized in the frequency domain. An inverse 3D FFT is applied on h to it to H. An example of the MACH filter is shown in Figure 2.4. Classification is based on analysing the responses from all such filters when a test video is applied as input to the filters.

Figure 2.4: Action MACH filter
Top row shows frames from 'Jumping Jack'sequence from [30] and corresponding Action MACH , Bottom row shows frames from 'Wave2'testing sequence followed by normalized correlation response against the 'Wave2'Action MACH filter. Adapted from [31]

Optical Flow

The above mentioned approaches capture motion by aggregating extracted human shape in some form from all the image frames in the sequence. Another powerful indicator of the motion in a sequence is *Optical Flow*. We give the details of optical flow and how it can be estimated. Optical flow is the apparent motion of brightness patterns in the image. Under constant illumination conditions, optical flow gives the motion pattern of an object of interest from a sequence of images. Dense or sparse optical flow methods from [32] [33] are often employed in applications. Given two successive frames, estimating the apparent motion between them is based on three assumptions (1) Brightness constancy - the intensity of a given world point on the image plane is same in all the frames of a sequence (2)Object motion between frames is small (3) Spatial Coherence - the intensity of a pixel is same as its neighbouring pixels. To obtain the displacement of a pixel intensity between two frames, let $I(x, y, t-1)$ and $I(x', y', t)$ be the intensities of (x, y) at time $t-1$ and (x', y') at time t respectively such that $x' = x + u$ and $y' = y + v$. According to brightness constancy assumption, we have, $I(x, y, t-1) = I(x', y', t) = I(x + u, y + v, t)$. Expanding as Taylors series, $I(x + u, y + v, t) \approx I(x, y, t) + I_x.u + I_y.v + I_t$ where $I_x \equiv \frac{\partial I}{\partial x}$ and $I_y \equiv \frac{\partial I}{\partial y}$. Hence

$$I_x.u + I_y.v + I_t = \nabla I \cdot \mathbf{u} + I_t = 0 \tag{2.1}$$

The displacement vector $\mathbf{u} = [u; v]$ (two unknowns) cannot be computed with the single known Equation 2.1. This is known as Horn-Schunck equation which holds for all the pixels in the image frame. The intuition behind this equation is that the component of the flow perpendicular to the gradient (i.e., parallel to the edge)

is unknown. This is known as the *aperture* problem. To overcome this most of the algorithms impose additional constraints like the spatial coherence proposed by Lucas and Kanade in [33]. Instead of trying to solve Equation 2.1 directly on all pixels, a small neighbourhood $N = n \times n$ around a pixel is considered. This results in an overdetermined system of n^2 equations and only 2 unknowns. For each pixel p_i in the neighbourhood N we can write

$$\nabla I(p_i) \cdot \mathbf{u} + I_t(p_i) = 0 \tag{2.2}$$

Now considering all pixels in the neighbourhood, $\mathbf{u}$ can be determined by minimizing the least squares problem $\Psi(\mathbf{u})$

$$\Psi(\mathbf{u}) = \sum_{p_i \in N} [\nabla I(p_i) \cdot \mathbf{u} + I_t(p_i)]^2 \tag{2.3}$$

Defining $\mathbf{A} \in \mathbf{R^{n^2 \times 2}}$ and $\mathbf{b} \in \mathbf{R^{n^2}}$ such that

$$\mathbf{A} = \begin{bmatrix} \nabla I(p_1)^T \\ \nabla I(p_2)^T \\ \nabla I(p_3)^T \\ .. \\ \nabla I(p_{n^2})^T \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ I_t(p_3) \\ .. \\ I_t(p_{n^2}) \end{bmatrix} \tag{2.4}$$

then the linear system in the matrix form would be $\mathbf{Au} = \mathbf{b}$. The solution $\mathbf{u} = \mathbf{A}^+\mathbf{b} = (\mathbf{A^T A})^{-1}\mathbf{A^T b}$ where $\mathbf{A}^+$ is known as *pseudoinverse* of $\mathbf{A}$. The solution to this linear system is unique only if the rank of $(\mathbf{A^T A})$ is 2.

The authors of [1] propose the use of optical flow to recognize actions from a distance implying on low resolution videos. Dense optical flow is computed on the figure centric frames. Optical flow is considered as a spatial pattern of noisy measurements instead of exact pixel displacements. The computed optical flow is rectified into different channels and each channel is filtered with a Gaussian to obtain a smooth spatial pattern. Each video is represented with a set of such patterns. Frame to frame similarity matrices are obtained by comparing individual frames in all channels between any test video and the training video whose identity is known. Final similarity matrix is obtained by summing up individual frame matrices by convolving with an identity matrix. Figure 2.5 shows the optical flow of a frame and frame to frame similarity matrix, identity and final similarity matrix. The approach was successfully applied to ballet, tennis and football datasets.

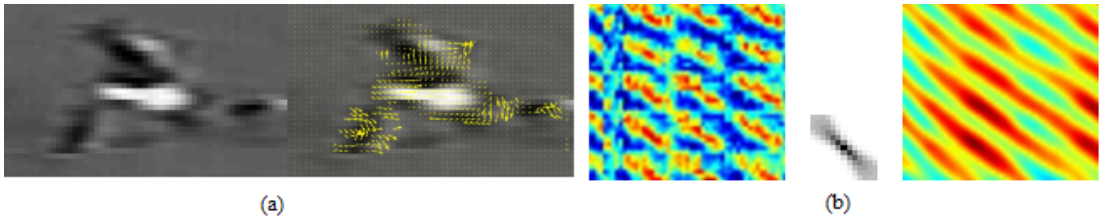

Figure 2.5: Computation of motion descriptors in [1]
(a)A figure centric frame and corresponding optical flow (b) Frame similarity matrices, Identity matrix, Final Similarity matrix.

2.1.2 Patch or grid based features

Global representations have been applied extensively and were successful in works mentioned above. However, if the representation relies only on the silhouette, the immediate drawback would be losing any local motion happening within the silhouette. Also these methods require efficient background subtraction methods. These drawbacks can be overcome to a certain extent if the *region of interest* ROI (human) is divided into patches or grids. Analysing each grid would give local appearance and motion information. All the grids collectively result in a global representation again. These representations require local matching of all the patches or grids for classification or recognition.

In [34], a video is represented using motion flows extracted from small 3-D patches of the video. 3D patches are extracted around every location $(x; y; t)$ of the volume. This captures a specific local motion within the patch. Given a test video, the similarity is measured by computing a space-time volume correlation between the template patches and patches from the test video. Scores are obtained at each location in the test video based on the local patch correlation. The system has to search for all possible 3D patches using sliding windows for the scores over the video.

Histogram of Oriented Gradients - HOG

A popular grid based representation, based on gradients was proposed by [35] for human detection and was extended to recognition in [36]. This representation or descriptor is the Histogram of Oriented Gradients (HOG). A human or an object can be detected by finding edges within an image. This is done by computing the gradients within the image at every pixel. Such detection eliminates the need of background subtraction methods. While optical flow can discriminate only moving objects, gradients can distinguish between moving and stationary objects.

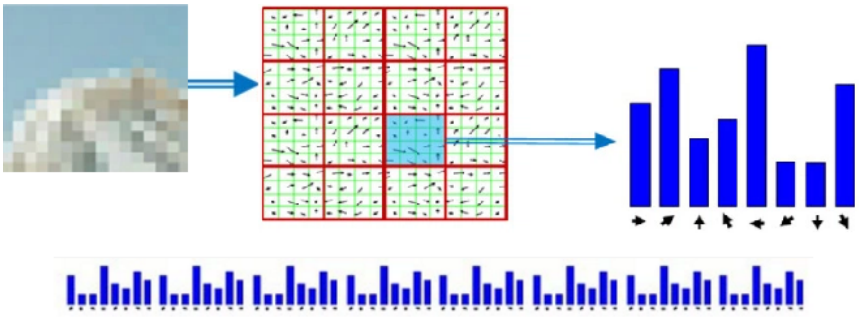

Figure 2.6: HOG features

Top row shows an image patch and the extraction of orientations from each cell followed by binning. Bottom image is the concatenation of histogram descriptors from all patches from the image [37]

For computing HOG, the image is divided into small patches or *cells*. In each cell,gradient directions or edge orientations are computed for each pixel and are accumulated according to the directions (binning). This gives a 1D histogram representation for each cell. Combining all these cell representations will result in a HOG descriptor for that image. Figure 2.6 illustrates the HOG descriptor. The local responses (from top row of Figure 2.6 are contrast normalized to correct the effects of illumination, shadowing. Normalization is done with respect to the histogram energy of larger spatial regions (groups of cells). In [35], after obtaining the HOG descriptor, a linear Support Vector Machine (SVM) is trained on examples from human and not human images for detecting humans in a given test image.

HOG3D is an extension to HOG proposed by Klaser et al [2]. HOG3D, proposed for action recognition is the histogram of three dimensional orientations of the gradients of a spatio-temporal cube. Figure 2.7 shows the computation of the descriptor. The cubes are extracted at different spatial and temporal scales. Each cube $\mathbf{c_i}$ is further divided into small blocks $\mathbf{b_j}$. These small blocks are over which the histogram $\mathbf{h_i}$ is computed. For each block mean of gradient orientations in that block is computed and is quantized using a regular polyhedron. The descriptor for each cube $\mathbf{c_i}$ is concatenation of the histograms of all the blocks $\mathbf{b_j}$.

Histograms of Optical Flow - HOF

Another histogram based representation are the Histogram of Optical Flow (HOF) features. The computation of the HOF is same as that of HOG, but optical flow orientations are used instead in forming the histogram. HOG features can rep-

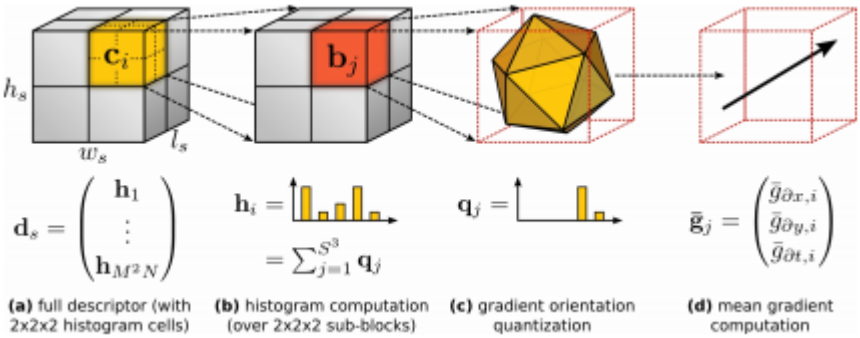

Figure 2.7: Illustration of the computation of HOG3D descriptors from [2]

resent local appearance and HOF features capture local motion within a spatio-temporal patch. Both HOG and HOF are combined for action recognition problem in [36]. *Motion Boundary Histograms* MBH were proposed in [38] for detecting humans from videos. Motion Boundary Histograms (MBH) is formed by the histograms of differential flow. Differential flow is the derivative of the image flow which eliminates any camera motion which varies smoothly on the image flow.

2.1.3 Interest Point based local features

Interest points are the local points in a video around which sudden changes in the appearance or motion are most likely to happen. Detecting these points and computing the descriptors around or at these points for action recognition was proposed in [39]. Harris corner detector [40] was extended to Harris 3D detector to extract Spatio Temporal Interest Points (STIP) in videos. Representing a video using descriptors at STIP is compact when compared with the earlier representations. The detector looks for spatio-temporal points where large changes in the intensity occur in the neighbourhood. For a given video, first a scale space is constructed by convolving the sequence with a Gaussian kernel. A spatio-temporal second moment matrix μ is constructed using the first order spatial and temporal derivatives of averaged with a gaussian weighting function. The interest points detected are based on the eigenvalues of μ. Classification is based on local neighbourhood comparison of the detected points. The same detector is used in [27] and [36] but with different descriptors at the interest points. In [27], and SVM is used for classification of events for the same descriptors in [39]. HOG and HOF are computed at the interest points and concatenate them as descriptors in [36] and classification is performed using multi-channel non-linear SVM. Figure 2.8 shows the detected interest points on the 'walking'event.

Interest points are detected as local maxima of a response function of the form

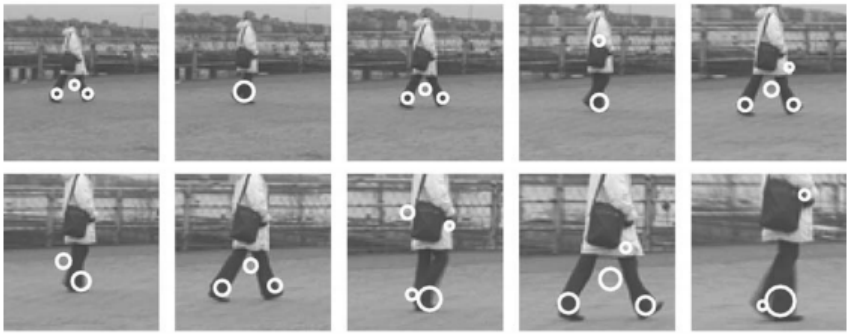

Figure 2.8: Spatio-temporal interest points
STIPs detected by the proposed Harris 3D detector on event 'walking'sequence in [39]

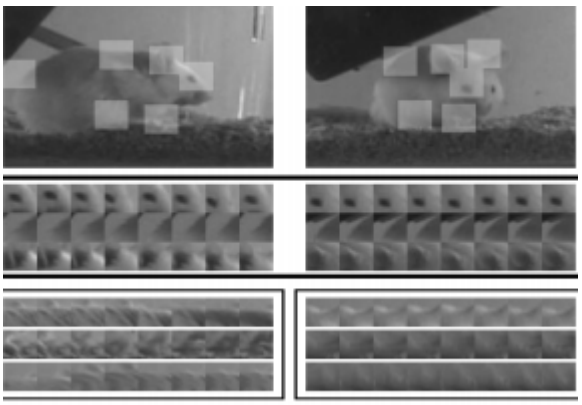

Figure 2.9: Cuboid descriptors of a sequence
Figure from [41] . Top row showing the interest points from two sequences of 'grooming'Other rows are stretched out cuboid descriptors. The similarity of 3 out of 6 cuboids is seen here.

$R = (I * g * h_{ev})^2) + (I * g * h_{od})^2)$ in [41] for behaviour recognition. $g(x, y, \sigma)$ is the 2D Gaussian kernel and h_{ev} and h_{od} are quadrature pair of Gabor filters. At each interest point *cuboids* (spatio-temporal pixels) are extracted as descriptors. Transforms like normalized pixel values, brightness gradient and windowed optical flow are applied on the cuboids to get various descriptor types. The classification strategy employed was similar to that of [1] which is based on computing the similarity matrix.

Hessian detector from [42] measures the saliency of a point using the determinant of the three dimensional Hessian matrix. Interest points detected are scale invariant using Hessian detector. Several other interest point detectors exist in the literature whose performance evaluation can be found in [43]. The evaluation is done on various combinations of detectors and descriptors at different space and time scales. The conclusion of this evaluation was that the *dense sampling* outperforms all the standard interest point detectors. Dense sampling is regular

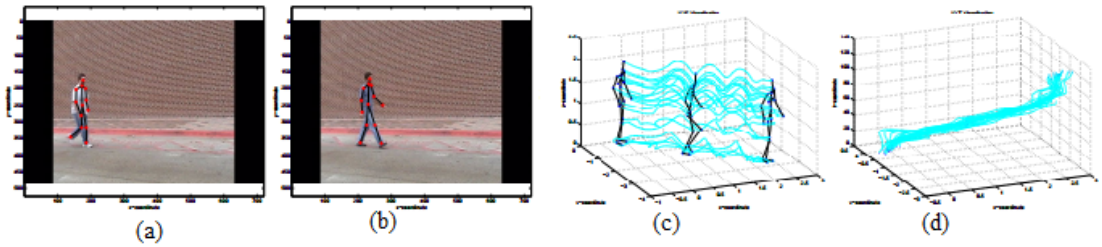

Figure 2.10: Trajectory descriptors for action 'walking'
(a) and (b) show the tracked joints on the frames for 'walking'sequences (c) and (d) show the representation of action 'walking'as trajectories in XYZ and XYT spaces respectively. Adapted from [46]

sampling of small video cubes $(x, y, t, \sigma, \delta)$ where σ and δ are the spatial and temporal scales respectively. This sampling is done with 50% overlap in space and time. The features from dense sampling are approximately 20 times more than those from other detectors because each patch sampled would result in constant number of features [44] .

2.1.4 Trajectories

Human motion in a video can also be represented by a set of points which create a path in the sequence. These paths are known as *trajectories* and the points which describe the path are generally Two dimensional (2D), 3D or Four dimensional (4D) representation of human joints. The early work of Johansson et al [45] on human motion perception has motivated research in this direction. It was suggested that humans can perceive motion by tracking joint positions. An action is represented as a set of 13 joint trajectories in 4D XYZT space in [46]. View-invariance on the trajectories is imposed by an affine projection, obtaining normalized XYT trajectories. A similar approach is used in [13]. Epipolar geometry of static images from stationary cameras is extended to geometry of dynamic scenes from moving cameras to handle the view-point variance. Figure 2.10 shows trajectories of joints in XYT and XYZ spaces. The trajectories shown are the paths traced by each of 13 human joints.

An extension to trajectory based approach is made by including a set of features computed over the trajectories along with the path traced. Recent work of Wang et al, [47] proposed the use of *dense trajectories* and features over these trajectories for action recognition. These trajectories are obtained by tracking densely sampled points over sequence at different spatial and temporal scales. HOG, HOF, MBH and trajectory descriptors are computed over the a small neighbourhood $N \times N$ along the trajectory and are combined as features. A non-linear SVM

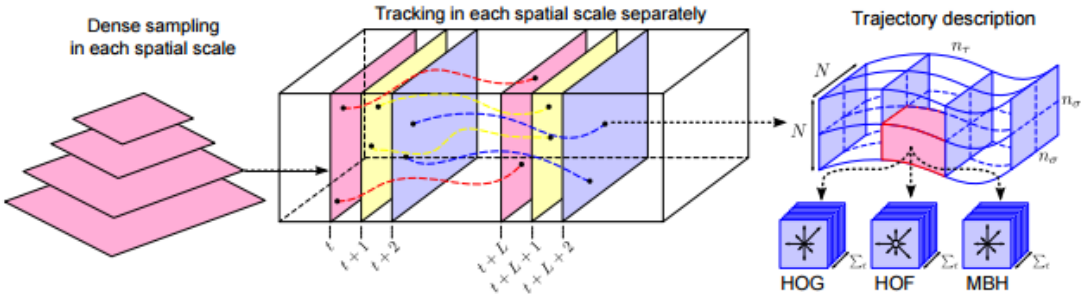

Figure 2.11: Dense trajectories feature extraction
Adapted from [47]

is trained on these features and is used for classification. Figure 2.11 illustrates the dense trajectory feature extraction.

2.1.5 Encoding methods

Previous sections have dealt with different methods of representing a given video based on global and local features. These representations are high dimensional and often the representations from local features of any two videos (even though of same class) would not be of same size. To overcome this local features are encoded such that the dimensions are reduced and are uniform over the entire dataset of videos. Among various encoding schemes [48] [16], Bag of visual words and Fisher vector encoding are state-of-the-art.

Bag of Visual Words - BoVW or BoW

BoW model from natural language processing is extended to Bag of Visual Words (BoVW) model for vision based applications. The main steps in encoding using this model is local feature extraction , *codebook* generation, encoding, pooling and normalization. Figure 2.12 shows the pipeline of the BoVW model. Codebook contains the *vocabulary* using which the data (text,image or video) is encoded. To generate a codebook, descriptors are sampled randomly over the entire training set. The sampled descriptors are clustered using *k-means* algorithm [49] into k clusters. The centers of these clusters are known as *codewords*. Each local descriptor is assigned to one of these clusters either by hard assignment methods or soft assignments like Vector Quantization (VQ) or sparse coding. To represent entire video as a single entity, local pooling is done which makes the local features robust to small translations. Average or max pooling is normally employed. The pooled features are then normalized either by $\|l\|_1$ or $\|l\|_2$ or power normalization.

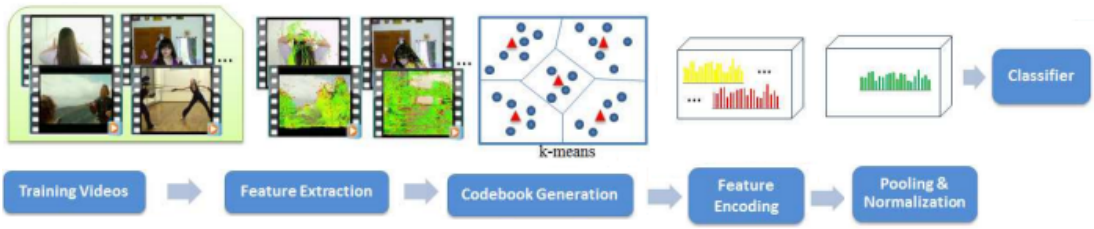

Figure 2.12: Bag of Visual Words model for action recognition
Adapted from [16]

BoVW model has been applied to action recognition successfully in [50] [27] [51].

Fisher vector - FV - encoding

Another method which has outperformed BoVW model is Fisher vector (FV) encoding method [15] [52]. This method follows similar pipeline as BoVW model. A Gaussian mixture model (GMM) is learnt on the training data instead of a codebook. The GMM is parameterized with *means* $\mu_{\mathbf{k}}$, *variances* $\rho_{\mathbf{k}}$ and *priors* $\mathbf{p_k}$ for each of k gaussians in the model. FV encoding captures the first and second order statistics between the local descriptors and the GMM. The encoded descriptors are further normalized by l_2 or power normalization. Dense trajectory descriptors with FV are the state of the art features in action recognition. It was shown in [53] that learning a kernel classifier using a Fisher kernel is same as learning a linear classifier on the FV encoded features. The complete details are given in Section 4.4.3 where this method is used for feature extraction.

2.1.6 Classification

Having now reviewed various representations of videos for action recognition from literature, we now move to the next phase which is classification. Classification is the task of assigning a 'label'to an unknown test data based on the information from known training data such that the loss in misclassification is minimized. When dealing with video data, classification methods are divided into those which act on *global* representation of an entire video sequence, known as *direct* methods, and those which consider the temporal evolution of a video sequence, known as *state-space* methods [25]. Our work is related to the direct methods of classification within a *supervised learning* scenario and hence the same are discussed in the following.

Nearest Neighbour Classifier - NN or k-NN

Nearest Neighbour (NN) or k-NN classfiers are non-parametric classifiers in pattern recognition . A test video is assigned the class label of that training video representation which is closest in some distance metric in the feature space.

Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3) \ . \ . \ . \ (\mathbf{x}_n, y_n)\}$ be the training set of features $\mathbf{x}_i \in R^d$ from n samples and corresponding labels $y_i \in R$. According to NN rule, the label y of a test sample $\mathbf{x}$ is given by y_{NN} of $\mathbf{x}_{NN}$ such that $\mathbf{x}_{NN} = \underset{\forall \mathbf{x}_i \in S}{\operatorname{argmin}}\{d_i(\mathbf{x}, \mathbf{x}_i)\}$. Typically, $d_i(\mathbf{x}, \mathbf{x}_i)$ is the Euclidean distance metric. NN error rate P is bounded by $P^* \leq P \leq P^*(2 - \frac{c}{c-1}P^*)$ as the number of training samples tends to infinity, where, P^* is the Bayes error rate and c is the number of classes [54]. It is extended to k- nearest neighbour (k-NN) by considering *'k'* closest neighbours instead of one. Intuitively, the decision boundaries are piecewise linear around each data point in the feature space if it is 1-NN classifier and around k subsets of training data for a k-NN classifier.

NN was used in [29], [30], [31], and more for classifying actions. Distance metrics other than Euclidean were also employed with action classification. For example, Hu moments with Mahalanobis distance are used in [29]. In [31] a prototype is built for each class encapsulating the inter-class variance as mentioned in Section 2.1.1. Each test is compared to the class prototypes rather than each training sample, within the NN classification scenario. A learned distance metric can be employed instead of a fixed one as in [55]. A discriminative key-pose frame is selected from the video sequence as a representative without any temporal ordering in [56]. These key-pose frames are used with the minimum distance NN classifier instead of whole video representations. NN classifiers are easy to implement because they are non-parametric. Although this classifier does not have a specific training phase, testing is slow as each test sample is compared to all the training samples. NN classifier is also sensitive to noise and outliers.

Support Vector Machines - SVM

SVM is a parametric classifier whose aim is to find an optimal hyperplane (decision boundary given by $\mathbf{w}^T x + b$) which maximizes the margin in the training data [57]. Margin is the distance from a hyperplane to the nearest training data point given by $2\frac{\mathbf{w}}{\|\mathbf{w}\|}$, where $\mathbf{w}$ is the normal vector to the hyperplane and b is the bias of the hyperplane. The parameters are learned from the training data during training phase. Testing will be with respect to the learned parameters. The SVM

binary classifier is learnt by minimizing the following optimization problem.

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i}^{n} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \tag{2.5}$$

where $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3) \ldots (\mathbf{x}_n, y_n)\}$ is the training set of features $\mathbf{x}_i \in R^d$ from n samples and corresponding labels $y_i \in \{+1, -1\}$. The solution of Equation 2.5 can be represented as linear combination of the training data given by $\mathbf{w} = \sum_{i}^{n} \alpha_i y_i \mathbf{x}_i$, where α are some coefficients. Equation 2.5 is primal form of the SVM optimization problem but often the dual in terms of α is considered shown in Equation 2.6.

$$\begin{aligned} &\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2}\sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j^T \mathbf{x}_k) \\ &\text{subject to } 0 \leq \alpha_i \leq C \text{ for } \forall i, \text{ and } \sum_i \alpha_i y_i = 0 \end{aligned} \tag{2.6}$$

In the above Equation 2.6, $\sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j^T \mathbf{x}_k) = \|\mathbf{w}\|^2 = \{\sum_j \alpha_j y_j \mathbf{x}_j\}^T \{\sum_k \alpha_k y_k \mathbf{x}_k\}$. The primal version of the classifier classifies a test data point $\mathbf{x}$ using $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ and the dual version of the classifier using $f(\mathbf{x}) = \sum_{i}^{n} \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b$. The main advantage of dual classifier is that most of α_i are 0 and hence Equation 2.6 is evaluated only at those i where $\alpha_i \neq 0$. Those corresponding $\mathbf{x}_i$ are known as *'support vectors'* and hence the name Support Vector Machine. When the data is not linearly separable in the feature space, using *kernel trick*, SVM learns a classifier in the kernel space. Kernel trick is to map the features to a high-dimensional space using some mapping functions called kernels. This leads to a non-linear SVM classifier. Figure 2.13 shows a typical hyperplane and support vectors for binary classification. The circled blue and red dots on either margins of the hyperplane are the support vectors. SVMs are extended to multi-class classification with (i) One-vs-All approach where a binary classifier is learnt for each class against the rest giving c classifiers for c classes and (ii) One-vs-One approach where binary classifiers are learnt for all pairs of classes giving $c(c-1)/2$ classifiers for c classes.

SVM is used for action recognition in [27] with local features from [39] and histogram features. A Gaussian kernel for local features and a χ^2 kernel for the histogram features is used to map the features into kernel space and the optimal hyperplane is learned in this space. It is extended to use with features from action videos with complex background in [27] and consistently performed better than

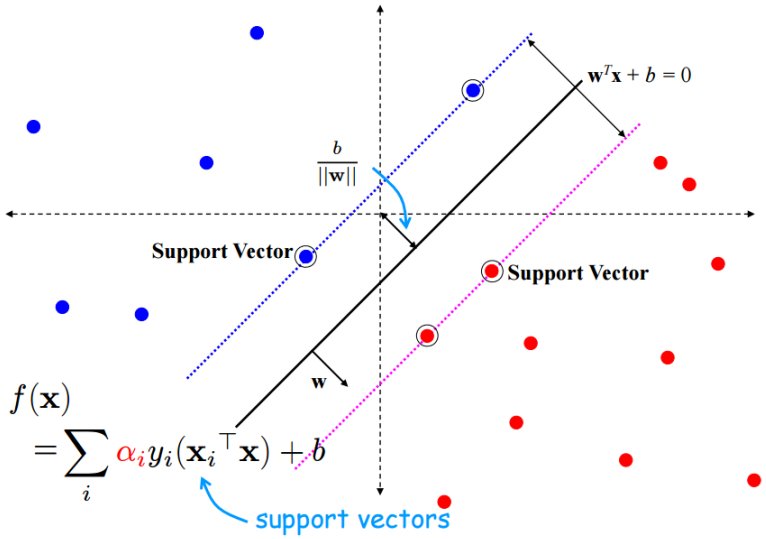

Figure 2.13: Support Vector Machine Classifier
Figure from [58] showing the hyperplane and support vectors.

NN classfier on the same set of features. Frame-based and video-based classification using SVMs is employed in [59]. Their system takes as input a sequence of frames of a video and outputs a vector, after performing a series of sampling and filtering operations. For video-based classification, final output vector from the system is considered as feature vector of a video and the output from a preceding stage is considered as feature vectors of corresponding frames.

2.2 Sparse representations

Most signals would have compact representations in specific domains, for example images in wavelet domain, sound in Fourier domain. This fact has lead to a new domain of signal representations popularly known as *sparse* representations. An exact sparse signal is the one which has very few non-zero values and zero elsewhere. An approximate sparse signal has most of the entries nearly zero. The basis in which a signal is sparse is termed as *dictionary* and the columns are known as *atoms*. Sparse representation or approximation is the problem of representing a signal as a linear combination of very few atoms from a dictionary. A detailed review on the theory of sparse representation is given by Bruckstein et. al. in [60]. Basic concepts given here are mostly from the same work and other works. Let $\mathbf{y}$ be any signal which has a sparse representation $\mathbf{x}$ in a dictionary $\mathbf{D}$. Solving the linear system of equations $\mathbf{y} = \mathbf{D}\,\mathbf{x}$, results in $\mathbf{x}$.

2.2.1 Problem Formulation

For most of the applications in Computer Vision, $\mathbf{D} \in \mathbf{R^{n \times m}}$ is *overcomplete* with $\mathbf{n} \ll \mathbf{m}$ and has infinite solutions. To mathematically formalize the problem, a measure of sparsity on $\mathbf{x}$ is imposed. As per the definition of sparse signal l_0 norm is introduced which is a count of number of nonzeros in $\mathbf{x}$. The problem can now be framed as Equation 2.7.

$$\min_{\mathbf{x}} \ \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\,\mathbf{x} \tag{2.7}$$

If $\mathbf{x}$ is known to be 'k'sparse (only s number of nonzeros and $s \ll m$) then the solution requires an exhaustive search over all 'k'subsets of $\mathbf{D}$. This is NP hard problem. Also l_0 is not a proper norm, which makes the problem non-convex. Another formulation is where l_0 is substituted with its closest convex function, l_1 norm, where $\|\mathbf{x}\|_\mathbf{1} = \sum_i |x_i|$. The convex relaxation of the problem can be stated as Equation 2.8.

$$\min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\,\mathbf{x} \tag{2.8}$$

2.2.2 Optimization

Equation 2.7 and Equation 2.8 aim to find the exact sparse solution $\mathbf{x}$. Often an error tolerance, ϵ is included in both formulations to account for any noise in the data resulting in the approximate formulations 2.9 and 2.10.

$$\min_{\mathbf{x}} \ \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{D}\,\mathbf{x}\|_\mathbf{2} \leqslant \epsilon \tag{2.9}$$

$$\min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{D}\,\mathbf{x}\|_\mathbf{2} \leqslant \epsilon \tag{2.10}$$

There are three different optimization approaches to solve these problems. Each approach is described in the following subsections.

Greedy methods

Solving the problem posed in Equation 2.7 or Equation 2.9 directly seems impossible. There are a class of methods or algorithms, known as Greedy Algorithms (GAs), optimize original problem subject to given constraints. GAs look for a locally optimal single-term update instead of doing an exhaustive search of subsets of the dictionary $\mathbf{D}$. It starts with an empty *support* of the solution $\mathbf{x}$ and expands with single column at every iteration such that the l_2 norm error in approximating $\mathbf{y}$ with the current support is minimized. The residual $\|\mathbf{y} - \mathbf{D}\,\mathbf{x}_k\|_\mathbf{2}$

(k is the current support) is computed with the new column. The algorithm terminates if the residual is less than a specific threshold. The algorithm commonly referred to as Orthogonal Matching Pursuit (OMP) [61] which is a modified version of Matching Pursuit (MP) in [62]. OMP is shown in Algorithm 1 .

For a dictionary size of mn and k_0 sparse $\mathbf{x}$, the number flops required by this

Algorithm 1: OMP Algorithm adapted from [60]

Input : Residual : $\mathbf{y}$
Dictionary : $\mathbf{D}$
threshold : δ
Output: Sparse Solution $\mathbf{x}$
Initialize: k = 0 and set
1. initial solution: $\mathbf{x^0} = \mathbf{0}$
2. initial residual: $\mathbf{r^0} = \mathbf{y}$
3. initial support: $\mathbf{S^0} = \emptyset$

Main Iteration: Increment k by 1 and do the following steps
Sweep: Compute the errors $\epsilon_j = \mathbf{min}_{z_j} \|\mathbf{d_j} z_j - \mathbf{r^k}\|_\mathbf{2}^\mathbf{2}$ for all j using the optimal choice $z_j^* = \mathbf{d_j^T r^{k-1}} / \|\mathbf{d_j}\|_\mathbf{2}^\mathbf{2}$
Update Support : Find a minimizer j_0 of $\epsilon(j) : \forall j \notin \mathbf{S}^{k-1}, \epsilon(j_0) \leq \epsilon(j)$ and update $S^k = S^{k-1} \bigcup j_0$
Update Provisional Solution : Compute $\mathbf{x^k}$, the minimizer of $\|\mathbf{y} - \mathbf{D}\, \mathbf{x}_k\|_\mathbf{2}$ subject to $Support\{\mathbf{x}\} = S^k$
Update Residual : Compute $\mathbf{r^k} = \mathbf{y} - \mathbf{D}\, \mathbf{x^k}$
Stopping Rule : If $\|\mathbf{r^k}\|_\mathbf{2} \leq \delta$, stop. Otherwise apply another iteration

would be $\mathcal{O}(k_0 mn)$ in general. Implying that if the algorithm outputs $\mathbf{x^k}$ sparse solution in k iterations. When compared to the exhaustive search, these are very fast. There are many variants of MP algorithms improvising in either complexity or accuracy or both. While this family of GAs mostly work and are extensively used, there are cases where these failed [63].

Convex methods

Another stratum of optimization methods are convex methods which aim at solving the convex versions, Equation 2.8 and Equation 2.10, of the actual problem. The problem is well defined now and can apply advanced optimization methods like interior-point methods or simplex methods. Since the problem is now convex, it can be solved for a global minimum. Convex methods are hence computation intense. The working of the algorithm can be analysed easily in this case. Most commonly these are known as *Basis Pursuit* BP algorithms [64]. The approximated problem in Equation 2.10 is termed as *Basis Pursuit Denoising* BPDN. It

was shown that minimal l_1 solution is also the sparsest solution in [65] and this has led to application of these methods to find the sparse solution $\mathbf{x}$ widely.

Iterated Shrinkage algorithms

Recently another line of optimization techniques emerged to solve the sparse solution problem, known as *Iterated shrinkage* methods [60]. These methods aim to solve the approximate convex problem, Equation 2.10, by iteratively shrinking the solution. The shrinkage is a $1D$ operation which sets to zero the entries which are less than a threshold. These methods iterate the multiplication step of $\mathbf{D}$ and its adjoint, followed by the shrinkage step until convergence. These methods were applied to image denoising, compressed sensing and tomography in [66].

2.2.3 Dictionary Design

The critical ingredient in solving for sparse representation is the dictionary. The uniqueness of the sparse solution is dependent on specific properties of the dictionary.

Properties of Dictionary

If a signal is known to have a sparse representation in a standard basis (unitary matrix), then it is unique. This comes from the orthogonality of the basis. This notion of orthogonality is extended as measures of coherence between the atoms or columns of overcomplete dictionaries which quantifies the uniqueness of the sparse solution obtained over that dictionary. These measures are (1) *Spark* (2) *Mutual Coherence* μ (3) Restricted Isometry Property (RIP).

Spark of a dictionary $\mathbf{D}$ is defined as the smallest number of linearly dependent columns in $\mathbf{D}$. The theorem on uniqueness of the sparse solution in terms of spark states that if a linear system $\mathbf{y} = \mathbf{D}\,\mathbf{x}$ has a sparse solution such that $\|\mathbf{x}\|_\mathbf{0} < spark(\mathbf{D})$, then that solution is the sparsest [60] . Spark is similar to rank of a matrix, but requires a combinatorial search of all possible subsets of $\mathbf{D}$. Computing the spark of a dictionary is as hard as solving Equation 2.7.

Mutual coherence μ of $\mathbf{D}$ is the largest absolute normalized inner product between different columns of $\mathbf{D}$. Let the k^{th} column of $\mathbf{D}$, then μ is given by Equation 2.11 .

$$\mu(\mathbf{D}) \quad = \quad \max_{\mathbf{1 \le k,j \le m, k \neq j}} \frac{|\ \mathbf{d_k^T d_j}\ |}{\|\mathbf{d_k}\|_\mathbf{2} \cdot \|\mathbf{d_j}\|_\mathbf{2}} \tag{2.11}$$

Mutual coherence or simply *coherence* is relatively simpler to compute than spark and characterizes the dependence of the columns of $\mathbf{D}$. μ must be as small as possible to ensure that $\mathbf{D}$ behaves close to a unitary matrix. In terms of μ, if a linear system $\mathbf{y} = \mathbf{D}\,\mathbf{x}$ has a sparse solution such that $\|\mathbf{x}\|_\mathbf{0} < (\frac{1}{2})(1 + 1/\mu(\mathbf{D}))$, then that solution is necessarily the sparsest.

Restricted Isometry Property (RIP) is an important criteria from [5] which is defined in terms of a constant $\delta_k < 1$ known as Restricted Isometry Constant (RIC) and an integer $k = 1, 2 \ldots$. A dictionary $\mathbf{D}$ obeys RIP of the order of k if each subset $\mathbf{D_k}$ formed by at most k columns of $\mathbf{D}$ has its nonzero singular values bounded above by $(\mathbf{1} + \delta_\mathbf{k})$ and below by $(\mathbf{1} - \delta_\mathbf{k})$. If D is a Gaussian random matrix then with high probability the following Equation 2.12 holds for all k sparse signals $\mathbf{x}$.

$$(\mathbf{1} - \delta_\mathbf{k})\|\mathbf{x}\|_\mathbf{2}^\mathbf{2} \leq \|\mathbf{D}\mathbf{x}\|_\mathbf{2}^\mathbf{2} \leq (\mathbf{1} + \delta_\mathbf{k})\|\mathbf{x}\|_\mathbf{2}^\mathbf{2} \tag{2.12}$$

Essentially RIP requires that every subset of k columns of $\mathbf{D}$ behaves like an orthonormal system.

The complexity of solving this system depends upon the nature of $\mathbf{D}$. In signal processing, the representation of the signal should be such that it expresses specific characteristics of the signal depending on the application besides being compact. For example, the representation must distinguish between signal and noise for denoising. For compression, most of the useful information must be captured in few coefficients. The representation must show distinct features for recognition and so on. Hence dictionary design is critical for an application in hand. The structure of dictionary has evolved over time [67] from a standard basis to more intelligent data-dependent one. Initially, the dictionaries were standard transforms like Fourier, wavelet, STFT etc. It was from seminal and influential works of [62] and [64], the term dictionary has overtaken the transforms. For the dictionary to represent varied features of the signal, union of bases was considered instead of a single transform. This has led to *overcomplete* or *redundant* dictionaries. Also, it was shown that sparse representation of a signal can be obtained in a more general setting where the basis(widely known as dictionary) is not strictly orthogonal under broad conditions [68].

Dictionaries can be categorised into *Analytic* and *Trained or Data-dependent* dictionaries. Analytic dictionaries are based on the mathematical model of the signal representation. Defined by a set of rules, these dictionaries are simple to construct and apply. Although fast, analytic dictionaries cannot capture the complex fea-

tures of natural signals.

Data dependent dictionaries

In computer vision and pattern recognition applications, the quest is for a representation which is more semantic than just being sparse. The dictionary in such cases must adapt itself to task in hand. Data dependent dictionaries have emerged in 1990s from the celebrated work of Olshausen and Field [69]. The work also emphasizes the connection between sparsity and human visual behaviour. Dictionary *learning* has become essential since then. The popular learning algorithms are Method of Optimal Directions (MOD) from [70] and K-Singular Value Decomposition (K-SVD) from [3]. Both these methods learn dictionaries alternating between two basic steps (1)Sparse coding step and (2)Dictionary update step. Starting with an initial dictionary normally constructed using random patches of data, sparse coefficients are computed. Using these sparse codes, mean squared error (mse) is computed over all the sparse codes of the data. The dictionary is updated with an update step such that the mse is minimized. The difference between MOD and K-SVD algorithm is in the dictionary update step. In K-SVD, a single atom is updated keeping all other atoms fixed, and the corresponding coefficients are also updated. For this reason, K-SVD is fast and is preferred over MOD. Another important type of data dependent dictionaries are those which are formed by directly using the features extracted from the data as atoms or columns [11]. These learned dictionaries are more suitable for computer vision applications and those which are formed from the data features directly are suited to classification problems.

2.3 Applications of Sparse Representation

Sparse representations find applications in statistics, signal processing and machine learning. Within the signal processing area, sparse approximations are applied in analysis, de-noising [3], inverse problems, compressive sensing of image, audio and video signals [60]. Depending on the application, a suitable variant of the problem is chosen. One of the core applications, *Image Denoising*, refers to problem of extracting a true image from its corrupted version. Let $\widetilde{\mathbf{y}}$ be an observed image which is assumed to be the mixture of original image $\mathbf{y}$ and additive white Gaussian noise $\mathbf{v}$ which means $\widetilde{\mathbf{y}} = \mathbf{y} + \mathbf{v}$. If $\|\mathbf{v}\|_\mathbf{2} \leq \epsilon$ then solving Equation 2.9 or its $\|l\|_1$ variant Basis Pursuit De-noising (BPDN), Equation 2.10, will result in a denoised image. Among many papers published on image denois-

(a) Original Image (b) Noisy Image (c) Denoised Image

Figure 2.14: Sample images showing denoising from sparse representations [3]

ing, the work by Elad and Aharon [3] stands out. A dictionary is trained on random image patches using K-SVD algorithm. A variant of Equation 2.9 with the trained dictionary is solved using OMP algorithm 1. Figure 2.14 shows some results from the same paper.

Another application *Image Inpainting* (or Interpolation) is the task of estimating the missing data of an observed image. An image to be inpainted is represented by a sparsifying dictionary. Fadili et al. in [71] impose a sparsity promoting prior penalty on the reconstructed coefficients and an Expectation Maximization (EM) method is proposed in a Bayesian network to recover the missing entries.

In Morphological Component Analysis (MCA), a signal $\mathbf{y}$ is considered as superposition of two or more subsignals $\mathbf{y_1}$ and $\mathbf{y_2}$ such that each of which has a sparse representation $\mathbf{x_1}$ and $\mathbf{x_2}$ in $\mathbf{D_1}$ and $\mathbf{D_2}$ respectively. In such cases, sparse representation problem Equation 2.9 can be modified to a source separation problem

$$\min_{\mathbf{x_1},\mathbf{x_2}} \|\mathbf{x_1}\|_0 + \|\mathbf{x_2}\|_0 \text{ subject to } \|\mathbf{y} - \mathbf{D_1}\,\mathbf{x_1} - \mathbf{D_2}\,\mathbf{x_2}\|_\mathbf{2}^\mathbf{2} \leqslant \epsilon_\mathbf{1}^\mathbf{2} + \epsilon_\mathbf{2}^\mathbf{2} \qquad (2.13)$$

This setting is useful in separating images into smooth cartoon part and texture part as in [72]. The dictionaries are chosen such that each image content is sparse in the respective one. BPDN with total variation regularization is employed to decompose the given image into smooth and texture parts and denoise simultaneously. From the same authors, MCA is extended for the application of inpainting in [73].

A dominant application of compressed sensing and sparse representations is the MRI [6, 74]. Magnetic Resonant Magnetic Resonant (MR) images, like angiograms are sparse in a corresponding transform domain. A single MR image is constructed by collecting a series of frames of data, called acquisitions. The data from this sequence of acquisitions are then used to reconstruct an image. Conventionally the data sampling rate is designed as per the Nyquist criterion, depending on the field of view. These two properties of MRI , sparse in trans-

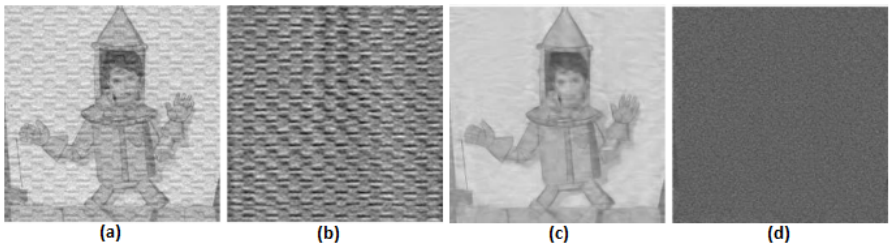

Figure 2.15: Morphological Component Analysis with Inpainting
Figure adapted from [72] (a)Image to be decomposed (b)Texture content (c) Smooth content (d) Noise

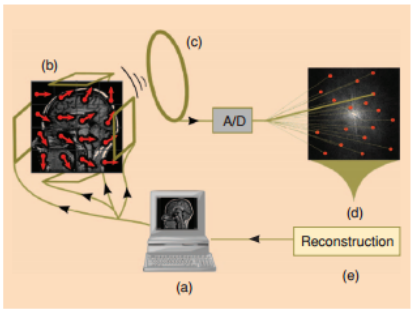

Figure 2.16: Compressed sensing based MRI system. Adapted from [74].

form domain and acquisition method, make MRI suitable for CS approaches [74]. Figure 2.16 shows the MRI system with CS mechanisms. The gradient of the image and the RF waveforms are controlled by the user. This controls the phase of the pixels/voxels in the image. An RF coil receives the signal in an encoded form. The incoherent measurements of the transformation space are obtained by meticulous tuning of the gradient waveforms. The image is reconstructed with nonlinear methods enforcing sparsity.

CS and classification are two potential areas where sparse representations are finding numerous applications. Applications where sparse representations are adopted for classification is elaborated in Section 2.4 and CS is introduced in the next Chapter 3.3.2 where it is more relevant.

2.4 Sparse representation based classification

Sparse representations were introduced to the classification domain by Huang and Aviyente [75] in 1998, but as a reconstruction tool. The dictionary over which the sparse representation is found is constructed with standard bases (Fourier for synthetic data and combination of Wavelet and Gabor for USPS hand-written dataset [76]) and the classification is obtained by combining Fisher discriminant term with the sparse approximation formulation. Another approach, *Supervised*

Dictionary Learning, was proposed in [77] where a single shared dictionary and decision functions for different classes are simultaneously learnt of for image classification. Variations of OMP and K-SVD were proposed in [12], for learning class dependent dictionaries. The learning process includes the quotient of the $\|l\|_2$ norms of the within classes and between classes scatter matrices for improving discriminative power of the representations. In [78], features for individual image groups are learned resulting in a discriminative representation of each image group. Each image is represented by either color histogram, BoW, HOG features or combination of all three. For each class of images, a dictionary is learnt from the training data. Reconstruction errors are computed for a given training set against all dictionaries. For each class, a distance measure known as importance vector is computed from the previously obtained reconstruction errors, which is distinct to that class. This importance vector would be used for deciding the class of the test image. Image classification in [79] is done with sparse coded Scale Invariant Feature Transform (SIFT) features and training a linear SVM.

The fact that the sparse representation itself encodes meaningful patterns was highlighted in the seminal work of Wright et al. [11] on face recognition. Their work relies on CS theory and sparse representations. It was shown that even random transformation of the data results in accurate classification, provided the sparse representation is computed properly. Yang et al. in [80] proposed sparse representation based distributed recognition and segmentation of human actions captured from a wearable motion sensor network. Albeit a sensor based model, this is the first published work on human action recognition using sparse representation. There are few works on action recognition with sparse representation with a vision based approach and these are reviewed in the next Chapter 3 where it would be more relevant.

Low rank approximations can be thought of as extensions to sparse representation problem and most of the terms and definitions are analogous. Hence the theory and related review is presented in Chapter 4 . Also the necessary background needed for deep feature extraction is introduced in Chapter 5 .

2.5 Summary

We can summarize some main points based on the review above. There are several factors which affect the accuracy of the system. The performance of any action recognition system relies predominantly on type of the features and the

classification strategy employed besides detection and tracking. The approaches either propose different features or classification methodology or both comparing with the state of the art. Also it can be affirmed that there is no generic method in either choosing the features or the classification method. Features depend on the type of the data being used and classification method to a certain extent depends on the features extracted.

The popular methods of classification in action recognition are Nearest Neighbour or its variations and SVMs with linear or non-linear kernels [27, 29, 30, 39] . NN method of classification assigns a label to a test sample based on a distance metric between the test and all the training samples in the feature space. This method does not consider any structure within the data or features. Basically its a one-to-one or one-to-few correspondence matching.

Classification with SVM requires an explicit training phase, so that the decision boundaries are learnt between the classes. It selects the decision boundaries based on the margin and the support vectors. In that again the method does not depend on the structure of the test data.

In the introduction Chapter 1, we have seen that in many applications of action recognition, data is redundant and asked if we can exploit the inherent structures of data. From the review on sparse representations, it is noted that the traditional classifiers could be replaced with sparse based classification which make use of *sparsity* of the data. There are two approaches of applying the sparse representations for classification, with or without dictionary learning. Classification with sparse representation has been explored exhaustively in applications dealing with images but is still an emerging trend with videos. Video data classification, based on these lines has witnessed affirmative results recently [17–19]. Motivated by this we further explore the sparse representations paradigm with compressed features for the problem of action recognition from videos in Chapter 3.

Another observation is that the classifier assigns the label to a single test data at a time. A novel method of classification is introduced in Chapter 4 based on the *low rank* structure of the data with which multiple test samples are classified as a single classification problem.

The feature extraction procedures in action recognition are normally followed by an encoding phase with BoVW or Fisher methods. Current state of the art features in action recognition are dense trajectory descriptors with fisher encoding [47]. The extraction of dense features is a computationally intense procedure and the dimensions of each descriptor even after encoding is quite large. In

Chapter 5, we propose deep features learnt automatically by a convolutional neural network which can be efficiently generated and are of moderate dimensions. These have proved to be better or equally well performance in our experiments.

Chapter 3

Sparse representation based Classification for Actions and Gestures

In this chapter, we present a solution to the problem of action and gesture recognition using sparse representations. The dictionary is modelled as a simple concatenation of features computed for each action or gesture class from the training data, and test data is classified by finding sparse representation of the test video features over this dictionary. Our method does not impose any explicit training procedure on the dictionary. Our approach is validated with two kinds of features, by projecting (i) Gait Energy Images (GEIs) and (ii) Motiondescriptors, to a lower dimension using random projection. Experiments have shown 100% recognition rate on standard datasets and are compared to the results obtained with widely used SVM classifier.

The work carried out in this chapter was presented at International Conference on Image Processing, Melbourne, 2013 [23].

3.1 Introduction

Sparse approximation techniques have found wide use in signal and image processing applications. Primarily developed for robust reconstruction of signals, sparse representations are currently adopted in classification problems where the goal is more than finding a compact representation. Though the idea that sparse representations can be used for signal classification was proposed in [75], the discriminative nature of sparse representation itself was not exploited until the published work of Wright et al [11]. Following this, sparse representations have evolved as classification tool, with most of the research on image classification [10,12,76,78,79,81]. In this work, sparse representation paradigm is explored further for classifying actions and gestures from videos where the main challenge posed is the large dimension of the data.

3.2 Related Work

In this section, related works which classify actions based on sparse representations are presented.

In [17], motion context descriptors are used as features which combine binary silhouette and optical flow from the frames to capture shape and motion. Binary silhouette and optical flow are extracted from a human-centric bounding box. These are histogrammed over 2×2 sub-windows of each bounding box. These are known as frame descriptors. Motion context is added to frame descriptor by considering a sequence of 15 frames centered with current frame. These are divided into 3 sets - past, current and future, of 5 frames each. For each set, corresponding frame descriptors are concatenated and using Principal Component Analysis PCA the dimension is reduced. This reduced context descriptor is appended to each frame descriptor to form a complete feature. Figure 3.1 illustrates the feature extraction procedure. For each frame in the test video the SRC algorithm is repeated and a label is given to that frame. The class of the test video is determined by the majority of the labels. The accuracy of this approach on Weizmann dataset was 96.77%.

In [18], Guo et al. propose logarithm of covariance matrix representation of a video sequence. A binary silhouette sequence is extracted for a given video sequence. This silhouette sequence is referred to as silhouette tunnel. The dynamics of the silhouette shape are captured throughout the tunnel by dense local feature descriptors. For each pixel belonging to the silhouette in the tunnel, a 13

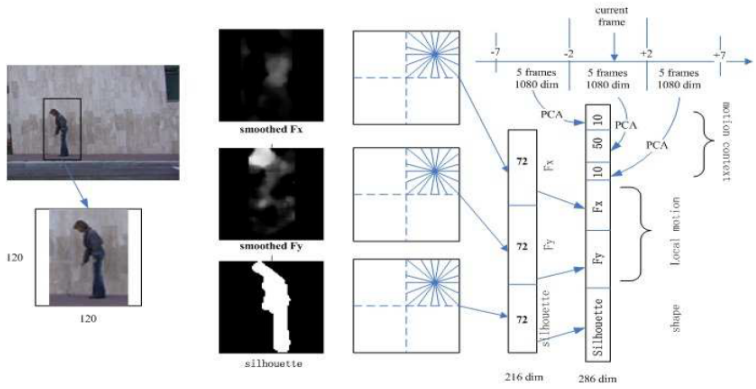

Figure 3.1: Motion Context Descriptors

Feature extraction from [17]. It illustrates the bounding box extraction followed by optical flow and binary silhouette within the bounding box. The final feature descriptor is also shown.

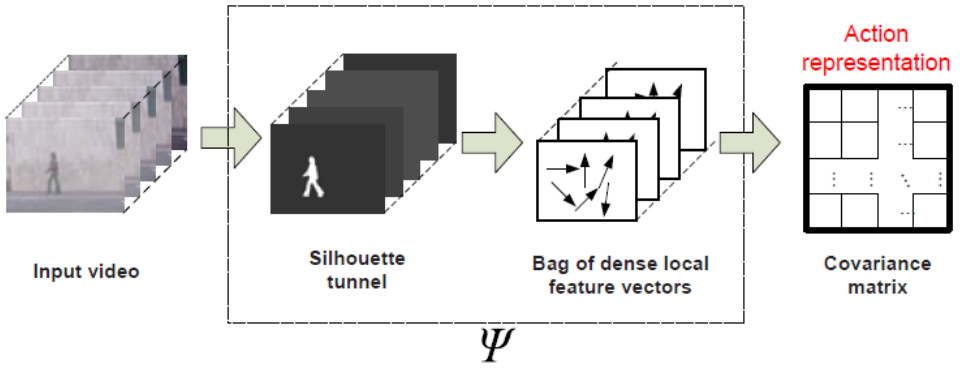

Figure 3.2: Computation of covariance descriptors

Figure shows the feature extraction procedure adopted in [18], starting from silhouette tunnel to the covariance matrix of the 13 dimensional feature descriptors.

dimensional feature vector is associated which includes the center (x, y, t) of the silhouette and Euclidean distances from the center to 10 different spatio-temporal directions. The mean of all the feature vectors for pixels throughout the silhouette is computed and a holistic representation of the video sequence is obtained by computing the covariance matrix from the feature vectors. The logarithm of the covariance matrices is computed and the obtained log-covariance features are used for classification. Their approach reported 96.74% accuracy on Weizmann dataset.

Dictionary learning with local features is proposed in [19]. These features are known as local motion pattern descriptors. Each video is divided into small segments temporally. Interest points are detected using 2D Harris detector [82] on the first frame of the segment. Spatio-temporal cubes are extracted around the

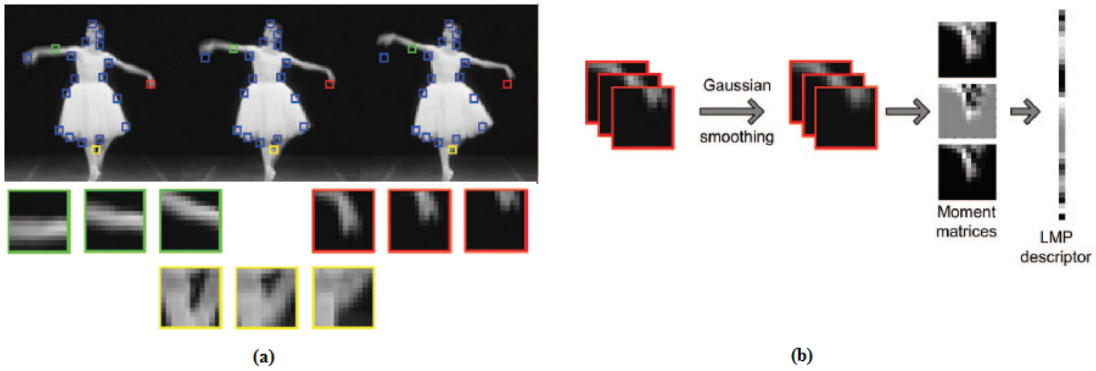

Figure 3.3: Local motion pattern extraction

LMP descriptor extraction from [19] (a) Detected interest points and three successive frames from 3 interest points (b) Formation of descriptor of a spatio-temporal cube at a specific detected interest point

detected points throughout the segment. The frames are pre-aligned with respect to the subject to track the local motion around a detected point. For each cube, mean is removed and variance, skewness and kurtosis are computed for each pixel along temporal direction. All these moments are concatenated to form a feature descriptor. Figure 3.3 shows the extraction of local motion pattern descriptors. Class-specific and shared dictionaries are trained using K-SVD algorithm on these features. Sparse representation is then computed over the learned dictionaries using OMP algorithm. The best performance on Weizmann dataset was reported as 98.9%

Though all the above works use sparse representations for classification, they differ in the type of features extracted and dictionary used. Except for [18], the other two methods of feature extraction are computationally intense and furthermore in [19] dictionaries are learnt on the encoded features. Also [18] requires robust background subtraction methods to extract the silhouette tunnel. Our approach shows that we can obtain comparative results with efficient (shape and optical flow) feature extraction methods without a trained dictionary.

3.3 Sparse representations for Action and Gesture Classification

We propose to classify actions and gestures based on sparse representation model within a supervised learning scenario. Mainly our approach is focussed on simple but robust dictionary construction using lower dimensional or *compressed* features. The general setting would be to construct the dictionary, say $\mathbf{A}$, using

features extracted from training data (videos with known action classes) and classify a test video by computing its sparse representation over $\mathbf{A}$. Let $\mathbf{y}_{ic}$ for $i = 1,\ 2,\ \ldots, k$ be k training samples in $c = 1,\ 2,\ \ldots, m$ classes each and let $\mathbf{A_m}$ be the set of training samples in m^{th} class, then $\mathbf{A_m} = [\mathbf{y}_{1m}\ \ \mathbf{y}_{2m}\ \ .\ .\ .\ \ \mathbf{y}_{km}]$. Including all m classes, the overall dictionary would be $\mathbf{A} = [\mathbf{A_1}\ \ \mathbf{A_2}\ \ .\ .\ .\ \ \mathbf{A_m}]$. A test or query video $\mathbf{y}_q$ can be classified by solving the system of linear equations given by Equation 3.1

$$\mathbf{y}_q = \mathbf{A}\,\mathbf{x}_q \tag{3.1}$$

where $\mathbf{x}_q$, the sparse representation of unknown $\mathbf{y}_q$, reveals the identity of $\mathbf{y}_q$.

3.3.1 Feature extraction

The most important information needed for action recognition are motion and appearance. Features for action recognition, broadly categorised as global and local features, capture this information in some form as reviewed in Chapter 1. Global features are computed taking into account the whole human figure in all the frames. Local features are computed at specific spatio-temporal points detected in the video by optimizing some saliency functions. In order to have complete motion pattern, we extract global features in this work. Specifically, we work with two types of features:

1. Gait Energy Images (GEIs)
2. Motion descriptors

The first step in computing these features is to extract foreground humans from all the frames in the video. Typically a mean-shift tracker is employed for this. The details of the complete feature extraction process are explained hereafter.

Mean-shift Tracking

Basically mean-shift is a non-parametric technique which seeks the local maxima of the underlying probability distribution function given some discrete sample from that function. The offset of a location $\mathbf{x}$ to a new location $\mathbf{x}'$ is obtained by $\mathbf{x}' = \mathbf{x} + \triangle_{\mathbf{x}}$ where $\triangle_{\mathbf{x}}$, the mean-shift vector is given by Equation 3.2

$$\triangle_{\mathbf{x}} = \frac{\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x})\mathbf{w}(\mathbf{a})(\mathbf{a}-\mathbf{x})}{\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x})\mathbf{w}(\mathbf{a})} \tag{3.2}$$

In equation 3.2, $\mathbf{x}$ is the current pixel, $\mathbf{a}$ are pixels in a local window around $\mathbf{x}$ and $\mathbf{w}(\mathbf{a})$ are the sample weights, $\mathbf{K}$ is kernel with non-negative, non-increasing

Figure 3.4: Illustration of GEI extraction
Figure shows the GEI extraction processes starting with a set of RGB frames to a single GEI representing that set of frames.

and piecewise continuous profile, for example a *Gaussian* or *uniform* kernel. The algorithm consists of three basic steps (1) compute an offset (the window) around each data point (2) calculate the mean of the data within the window (3) centre the window to the mean and repeat until convergence. Choosing the right kernel scale is critical for tracking. A large kernel window would confuse the algorithm with the background clutter and might converge on more than one mode and a smaller kernel window randomly searches on a likelihood region around the mode. There is no sound method of choosing and adapting the kernel scale in the algorithm. Besides this limitation, the sample weights $\mathbf{w}(\mathbf{a})$ also must be non-negative. Mean-shift blob tracker through scale space [83] addresses these issues. Modifying the mean-shift vector as in Equation 3.3, non-negative sample weights limitation is fixed.

$$\triangle_{\mathbf{x}} = \frac{\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x})\mathbf{w}(\mathbf{a})(\mathbf{a}-\mathbf{x})}{|\sum_{\mathbf{a}} \mathbf{K}(\mathbf{a}-\mathbf{x})\mathbf{w}(\mathbf{a})|} \tag{3.3}$$

A filter bank of spatial Difference of Gaussian (DoG) filters are convolved with the sample weight image to generate a scale space. These results are again convolved with an Epanichikov kernel in the scale dimension. Since these convolutions are computation intensive, the relationship between shadow kernels and mean-shift kernels is exploited to speed up the algorithm. The modified algorithm is a two-stage interleaved mean-shift procedure between spatial and scale modes, seeking local mode in the scale space resulting in a stable and natural selection of the kernel size. This tracker algorithm [83] is used in this work to extract foreground humans from background.

Gait Energy Images - GEIs

Gait Energy Image (GEI)s were proposed for recognizing individuals from their gait style [84]. The output from the tracker is converted to a binary silhouette.

Figure 3.5: Input frames for computing motion descriptors in [1]
The perfect alignment of these frames [1] can be noted from the above figure.

The GEI for each individual is computed by taking the average of binary silhouettes of that individual from all frames in that video. Let $\mathbf{I_b}(\mathbf{x}, \mathbf{y}, \mathbf{t})$ be the sequence of N binary silhouettes from a video. The GEI $\mathbf{I_g}(\mathbf{x}, \mathbf{y})$ is given by 3.4 and Figure 3.4 gives an idea of GEI extraction.

$$\mathbf{I_g}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{t=1}}^{\mathbf{N}} \mathbf{I_b}(\mathbf{x}, \mathbf{y}) \tag{3.4}$$

Essentially developed for recognition of gaits, GEIs can be used for action recognition on less complicated datasets. The efficiency of GEIs depends on the pre-alignment of the corresponding silhouette frames which in turn depend on the tracker output alignment.

Motion Descriptors

Motion descriptors are optical flow based descriptors which were used to recognize actions from a very low resolution sports video footage [1]. Optical flow is computed between two successive frames, rectified into 4 channels, 2 for each direction component and smoothed using a gaussian filter. Motion descriptors in [1] are extracted on aligned human-centric frames as shown in Figure 3.5.

There are two differences in the way the motion descriptors are computed in this work. First, the output from the tracker is not stabilized with respect to human centres as in [1]. Figure 3.6 shows the sample frames from the gesture dataset used in the experiments (more details on the dataset follow in Section 3.4.1). The reason for this is that there was no background data recorded for the dataset. A random frame from the sequence is input as the background to the tracker algorithm. Second, optical flow is computed in an overlapping manner. Instead of computing between every pair of successive frames, frames at some regular temporal offset are taken. For example, if there are N frames in a video sequence at t = *1,2, . . ., N* and the temporal offset is 'n'then every *(t,t+n)* frames

Figure 3.6: Sample frames from HWU dataset

Figure shows the frames of two gesture sequences from HWU gesture dataset used in our experiments. It is seen that the frames are not aligned perfectly and the background also is not completely static.

are considered for computing optical flow. The global motion between any two consecutive frames would be very small for the datasets used. If the optical flow is computed at regular offsets only then one might miss the local motion between the consecutive frames. Hence the overlapping scheme is adopted. The process is illustrated in Figure 3.7.

3.3.2 Random Projection - RP

The dimensionality of features obtained from video data will be enormous naturally. Efficient dimensionality reduction methods are indispensable to reduce the computational complexity while preserving the information encoded in the features. RP proved to be a competent dimension reduction method to conventional PCA [85]. According to Johnson-Lindenstrauss (JL) lemma, the distances between points in high-dimensional space are preserved when projected to much lower dimension using random projection. The term distance here is in the $\|l\|_2$ sense. Mathematically, if $\mathbf{\Phi}$ is a projection matrix in $\mathbf{R^{n \times N}}$ such that $\mathbf{n} \ll \mathbf{N}$ and its entries randomly drawn, and $\mathbf{y_1}$, $\mathbf{y_2}$ are two vectors in $\mathbf{R^N}$, it can be stated as Equation 3.5 with $\epsilon \in [0, 1]$ [86].

$$(\mathbf{1} - \epsilon)\|\mathbf{y_1} - \mathbf{y_2}\|_\mathbf{2}^\mathbf{2} \leq \|\mathbf{\Phi y_1} - \mathbf{\Phi y_2}\|_\mathbf{2}^\mathbf{2} \leq (\mathbf{1} + \epsilon)\|\mathbf{y_1} - \mathbf{y_2}\|_\mathbf{2}^\mathbf{2} \tag{3.5}$$

The following are examples of $\mathbf{\Phi}$:

(a) entries $\Phi_{i,j}$ drawn independently from a Gaussian distribution

$$\Phi_{i,j} \sim \mathcal{N}(0, \frac{1}{n}) \tag{3.6}$$

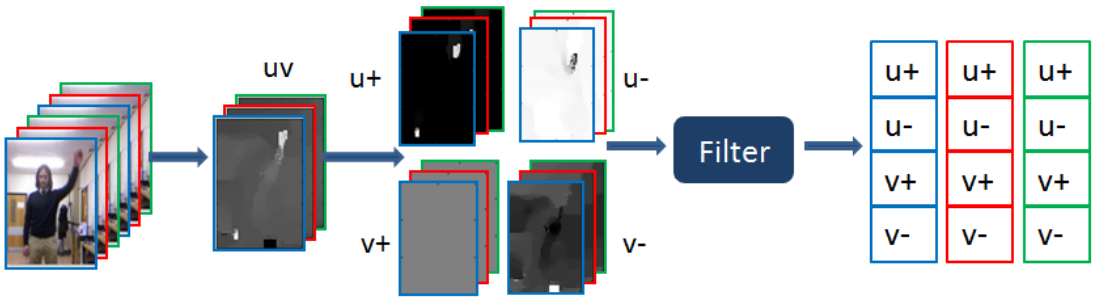

Figure 3.7: Motion descriptor extraction in our experiments

Figure shows the sequence of motion descriptor extraction starting with computation of optical flow to the formation of final descriptor. The colour coding is to show the correspondence from frames to the descriptors.

(b) entries $\Phi_{i,j}$ which are independent realizations of $\pm$ Bernoulli's distribution

$$\Phi_{i,j} := \begin{cases} +\frac{1}{\sqrt{n}} \text{ with probability } \frac{1}{2} \\ -\frac{1}{\sqrt{n}} \text{ with probability } \frac{1}{2} \end{cases} \tag{3.7}$$

(c) entries $\Phi_{i,j}$ drawn such that

$$\Phi_{i,j} := \begin{cases} +\sqrt{\frac{3}{n}} \text{ with probability } \frac{1}{6} \\ 0 \text{ with probability } \frac{2}{3} \\ -\sqrt{\frac{3}{n}} \text{ with probability } \frac{1}{6} \end{cases} \tag{3.8}$$

The relevance of JL lemma to sparse representations is explained next.

Compressed Sensing - CS

CS is an offshoot in the sparse signals domain which has gained immense interest recently. CS theory asserts that a signal can be recovered by sensing or sampling far fewer samples than the Nyquist criteria [5, 8]. The underpinning principles of CS are (i) the signal to be recovered is *sparse* in some basis and (ii) the sensing modality is *incoherent* to the sparsifying or the representation basis. Let $\mathbf{y}_c$ be the signal obtained by non-adaptively sampling few measurements of $\mathbf{y}$. Let $\mathbf{y}$ has a sparse representation $\mathbf{x}$ in $\mathbf{D}$ similar to the Equation 3.1. The recovery of $\mathbf{y}$ can be obtained by solving for sparse $\mathbf{x}$ in the CS problem given by

$$\mathbf{y}_c = \mathbf{\Phi D}\ \mathbf{x} \quad = \mathbf{A}\ \mathbf{x} \tag{3.9}$$

where $\mathbf{\Phi}$ is a random sensing matrix and $\mathbf{A} = [\mathbf{\Phi D}]$. For the recovery of sparse signals, it was mentioned in previous chapter that, the dictionaries must obey RIP. It is reproduced here for convenience. Let $\mathbf{x}$ be the sparse representation in $\mathbf{D}$ corresponding to $\mathbf{y}$. If $\mathbf{D}$ is a Gaussian random matrix then, RIP states that

$$(\mathbf{1} - \delta_{\mathbf{s}})\|\mathbf{x}\|_{\mathbf{2}}^{\mathbf{2}} \;\leq\; \|\mathbf{Dx}\|_{\mathbf{2}}^{\mathbf{2}} \;\leq\; (\mathbf{1} + \delta_{\mathbf{s}})\|\mathbf{x}\|_{\mathbf{2}}^{\mathbf{2}} \tag{3.10}$$

for $\delta_s < 1$. In order for the RIP of $\mathbf{A}$ to be preserved, $\mathbf{\Phi}$ must be of any one of the distributions as mentioned above in Equation 3.6 or Equation 3.7 or Equation 3.8 implying that all s subsets of columns of $\mathbf{A}$ are nearly orthogonal. That is, if $\mathbf{x_1}$ and $\mathbf{x_2}$ are two s sparse representations in $\mathbf{A}$ and $\delta_{2s} < 1$ then the distance between $\mathbf{x_1}$ and $\mathbf{x_2}$ are preserved in the measurement space as shown in Equation 3.11 .

$$(\mathbf{1} - \delta_{\mathbf{2s}})\|\mathbf{x_1} - \mathbf{x_2}\|_{\mathbf{2}}^{\mathbf{2}} \;\leq\; \|\mathbf{Ax_1} - \mathbf{Ax_2}\|_{\mathbf{2}}^{\mathbf{2}} \;\leq\; (\mathbf{1} + \delta_{\mathbf{2s}})\|\mathbf{x_1} - \mathbf{x_2}\|_{\mathbf{2}}^{\mathbf{2}} \tag{3.11}$$

This is similar to JL Equation 3.5, except now the signals are s sparse meaning that the signal is zero except for the highest s non-zeros. This shows the relevance of RIP of CS to JL lemma [86]. Now $\mathbf{A} = [\mathbf{\Phi D}]$ should be thought as the distance preserving matrix satisfying JL lemma. Also $\mathbf{A}$ in equation 3.1 is obtained from $[\mathbf{\Phi D}]$. Another property on the dictionary to recover a unique sparse solution is that the *Mutual coherence* μ must be smallest possible. The standard orthonormal bases like Fourier, DCT are *incoherent*. However, not all signals are sparse in single orthonormal basis. Instead they are sparse in concatenation of bases. This concatenation leads to *overcomplete* and *redundant* dictionaries and thereby incoherence and RIP are rarely satisfied by the dictionaries. Also the data dependent dictionaries in vision and pattern recognition applications are overcomplete and *coherent*. Since the goal of the CS problem is to recover $\mathbf{y}$ which is $\mathbf{Dx}$ (rather than a unique sparse $\mathbf{x}$), $\mathbf{D}$ satisfying incoherence is not really required [87]. There are efficient algorithms in the literature which solve Equation 3.9 and one of which is explained in Section 3.3.3.

The dimensionality of the features extracted in our work is reduced using random projection. The advantage of RP over other dimensionality reduction methods like PCA is that it is independent of data and can be generated efficiently. The drawback of data-dependent methods is that it requires re-computation of projection basis whenever there is an extension or modification of training data. Besides, these methods are computationally intense.

3.3.3 Classification

The dictionary is formed by arranging the obtained lower dimensional features from the training data as columns class-wise and classification of a test data is based on solving Equation 3.1 for $\mathbf{x}$. Implicitly we solve Equation 3.9 for random projected features. The core of classification is the computation of sparse solution over a given dictionary. The mathematical formulation for solving $\mathbf{x}$ is seen in Chapter 2, Section 2.2. We use convex relaxation of the problem where l_1 norm of the sparse solution is minimized instead of l_0 norm. Depending upon if each data sample is represented as feature vector or feature matrix, we use two methods of computing sparse solution. If $\mathbf{y}$ is feature vector, L1 regularised least squares (l1-ls) solver is used to compute $\mathbf{x}$. If $\mathbf{y}$ is a feature matrix then Multiple Basis Pursuit (M-BP) is used.

L1 regularised Least Squares - l1-ls Solver

l1-ls solver [88] was proposed to solve large scale l_1 norm minimization problems within the CS domain. These problems commonly occur with MRI applications. The optimization problem it solves is

$$\text{minimize} \quad \|\mathbf{Ax} - \mathbf{y}\|_{\mathbf{2}}^2 + \lambda\|\mathbf{x}\|_1 \tag{3.12}$$

where $\mathbf{A} \in \mathbf{R}^{n \times km} = [\mathbf{\Phi D}]$, n is the reduced dimension of the feature vector and km are the number of training samples from all m classes as per our formulation. Basically it is an interior-point optimization with preconditioned conjugate gradients (PCG) algorithm to compute the search direction.

Multiple Basis Pursuit - M-BP

M-BP algorithms [89] solve simultaneous sparse approximation problem of several signals over a dictionary. The least squares term in 3.12 is replaced with 'Frobenius'norm and the l_1 norm is replaced with *row sparsity* inducing penalization on sparse coefficient matrix. If $\mathbf{Y}$ is the matrix of signals, $\mathbf{A}$ is the dictionary as in Equation 3.12 and $\mathbf{X}$ is the sparse coefficient matrix, then the optimization problem it solves is

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{Y} - \mathbf{AX}\|_{\mathrm{F}}^2 + \lambda \sum_i \|\mathbf{x}_{i,.}\|_2 \tag{3.13}$$

where $\mathbf{x}_{i,\cdot}$ is the i^{th} row of the coefficient matrix $\mathbf{X}$. This is solved by iteratively shrinking the coefficient matrix until convergence.

Decoding the sparse solution for classification

The sparsest solution seeks the best basis subset in which it is concise. The sparse solution encodes the label of the test data in terms of its support, if it the solution is exact. The solution will not be exact but nearly sparse due to the mutual correlation between the classes. Nearly sparse means that most of its entries are nearly zero but not exactly zero. In such cases, class-wise residual errors are computed as in [11] and assign that class label to the test ($\mathbf{y}$) with minimum error. For this, construct a vector $\mathbf{x}_i$ for each class i, of same size as the sparse solution $\mathbf{x}$ and is equal to $\mathbf{x}$ only at indices corresponding to i^{th} class and zero elsewhere. The residual errors are given by

$$\mathbf{e}_i = \|\mathbf{y} - \mathbf{A}\mathbf{x}_i\|_\mathbf{2} \tag{3.14}$$

In case of simultaneous sparse approximation, $\mathbf{e}_i = \|\mathbf{Y} - \mathbf{A}\mathbf{X}_i\|_\mathbf{F}$. The predicted class of $\mathbf{y}$ or $\mathbf{Y} = \text{argmin}_i\, \mathbf{e}_i$. Algorithm 2 below summarizes our approach.

Algorithm 2: Classification using sparse representation adapted from [11]

Input : Test data $\mathbf{y}$ or $\mathbf{Y}$ and Dictionary $\mathbf{A}$
Output: Label i of Test data
Normalize the columns of $\mathbf{A}$ and test data to have unit norm
Solve for $\mathbf{x}$ using Equation 3.12 if test is $\mathbf{y}$
Solve for $\mathbf{X}$ using Equation 3.13 if test is $\mathbf{Y}$
Compute residual errors $\mathbf{e}_i$ using Equation 3.14 for each class $i = 1, 2, ..m$
Label($\mathbf{y}$) = $\text{argmin}_i\, \mathbf{e}_i$

3.4 Experiments

The proposed approach, Sparse representation based Classification (SRC) for classifying actions and gestures from videos using compressed features, is evaluated on the two datasets under various settings. Details of the datasets used in the experiments is given before we move onto evaluation strategies.

3.4.1 Datasets

We perform experiments on two datasets. First is on the widely known Weizmann dataset [30] of actions. This is a standard dataset used to assess the performance

Figure 3.8: Snapshot of Weizmann dataset of actions

Figure 3.9: Snapshot of HWU dataset of gestures

of any new methodology on action recognition. It comprises of 90 videos of low resolution (180 x 144, deinterlaced 50fps) with 9 subjects performing 10 actions individually. The actions include bend, jump, jump in place, jumping jack, run, skip,side-walk, walk, wave. A snapshot of the dataset is shown in Figure 3.8.

Second is HWU dataset of gestures [90]. This dataset was proposed by Baratteni et al. for the control of industrial collaborative robots. This is a gesture dataset consisting of reasonable resolution (640 x 420 with 30fps) 10 gestures performed by 5 or 6 actors. The gestures in the dataset are done, faster, follow-me, home, identification, interaction, ok, slower, start,stop. The data is recorded by Kinect. The recordings are available in 3 forms (i) Skeleton (ii) Depth frames and (iii)RGB frames. RGB images of the recordings are used in our experiments. Figure 3.9 shows the details of this dataset.

3.4.2 Evaluation

In all the experiments, performance is measured as the overall classification accuracy or recognition rate with Leave-One-Out-Cross-Validation (LOOCV) and is compared with the linear SVM classifier and NN Classifier.

Dictionary construction

The dictionary for computing sparse representations is constructed by concatenating the features from training data as columns, class-wise as described in Section 3.3. GEIs and motion descriptors are computed on Weizmann dataset of actions and motion descriptors on HWU gesture dataset. Computation of GEIs requires binary silhouettes which rely on background subtraction. For Weizmann dataset, the background frames are available and hence silhouettes are extracted efficiently. For HWU dataset, the background frames are not recorded and that

is the reason, only motion descriptors are extracted from HWU dataset. GEIs for Weizmann dataset are computed according to the illustration in Figure 3.4. The size of each GEI was slightly different and hence resized to 120 x 80. This gives a feature vector of dimension 9600 x 1. With GEIs as features and LOOCV (8 subjects for training and 1 as test), the dictionary size for the Weizmann dataset would be 9600 x 80 for 10 classes.

Motion descriptors are computed as illustrated in Figure 3.7 and Section 3.3.1. First, the tracker output frames are resized to a smaller size of 60 x 40 for both datasets, since determining optical flow is computation intense. The default algorithm of [91] is used to compute optical flow. After rectifying into four channels and filtering the size of each optical flow frame is four times the frame size (4 x 60 x 40) which is 9600. The length of each video is different due to different action types and repetitions and hence the number of frames is not same. In the experiments, 14 optical flow frames are considered for each video. Each video is represented as a 9600 x 14 matrix and the dictionary with the Weizmann training data would be 9600 x 1120 and 9600 x 560 for HWU gesture data (4 subjects in training and 1 as test).

Random Projection Vs Downsampling

Our first experiment highlights the significance of Random projection as the dimensionality reduction tool. This experiment is conducted on GEIs from the Weizmann dataset [1]. The dimensions are reduced by (a)DS as this also is a data-independent method and (b)RP, to form dictionaries of different sizes. The projected dimensions are 72, 128, 256, 512, 1024 and 2048 with RP and 70, 88, 96, 150, 280 and 2400 with DS. The entries of the random matrix for RP are drawn from $\mathcal{N}(0,1)$ and follow the same steps as in [85]. Let $\mathbf{y} \in \mathbf{R^N}$ be the feature with $N = 9600$, and $\mathbf{\Phi} \in \mathbf{R^{n \times N}}$ be the random matrix with n taking any of the values mentioned above. The low dimensional $\mathbf{y}_\mathrm{n}$ is used in the sparse representation model as $\mathbf{y}_\mathrm{n} = \mathbf{\Phi x}$. Sparse representation for each test subject is computed using l1-ls solver. The matlab toolbox libsvm [92] is used for SVM classifier. Table 3.1 and Table 3.2 show the performance comparison between these classifiers with DS and RP dimension reduction methods respectively. The average performances over different dimensions are tabulated in the Table 3.3.

[1]The results under this heading in our paper [23] are different from what we present here. We have noticed an improvement in the accuracies when the random matrix columns are also unit normalized before the projection. The updated results are presented here.

Projection	SRC (%)	SVM (%)	NN (%)
70	**95.56**	93.33	93.33
96	95.56	**96.67**	94.44
150	**96.67**	95.56	94.44
600	**97.78**	95.56	94.44
2400	**97.78**	94.44	94.44

Table 3.1: Comparison of the performances with reduced dimension using Down-sampling

Projection	SRC (%)	SVM (%)	NN (%)
72	**97.78**	96.67	96.67
128	**98.89**	96.67	95.56
256	**98.89**	94.44	94.44
512	**97.78**	**97.78**	94.44
1024	**97.78**	93.33	94.44

Table 3.2: Comparison of the performances with reduced dimension using Random Projection

GEIs Vs Motion Descriptors

To see how the choice of features effects for recognition rate with SRC, we compare the experiments with GEIs (shape-based features) from the previous section and motion descriptors (flow-based features) on Weizmann dataset. Each motion descriptor feature is projected to a lower dimension ($9600 \rightarrow 256$) using RP giving in a dictionary of size 256×1120 for motion descriptors. Each test sample is 256×14. Sparse solution is computed using Equation 3.13. M-BP algorithm is used to solve this. The recognition rate is increased to **100%** from 98.22% which was obtained with GEIs. With motion descriptors, SVM and NN also reach **100%** classification accuracy. For classification with SVM and NN, each sample 256×14 is stretched out as a vector. The recognition rate of **100%** for the Weizmann dataset was also obtained previously in [30, 93]. The reason for this is the simplicity of the dataset. Even these days, this dataset is still used as a starting point of validating a novel aspect in the field of action recognition. For the same reason, we have also used this dataset.

SRC Vs SVM and NN

To validate the robustness of SRC, the experiments are repeated on the HWU gesture dataset. Key differences to be observed from the Figures 3.9 and 3.8 are that the gesture dataset frames are not strictly human-centered (or perfectly aligned) and the background is not as static as that of Weizmann dataset. Motion

Projection	SRC (%)	SVM (%)	NN (%)
RP	**98.22**	95.78	95.11
DS	**96.67**	95.18	94.44

Table 3.3: Comparison of the performances with RP and DS
This shows the average accuracy over different dimensions for each classifier.

descriptors are extracted from the videos and the dimensions are reduced using RP . The reduced dimension is 256 and the dictionary size is 256×560. The test sample size is 256×14. The sparse solution is obtained in the same way as in Section 3.4.2 using M-BP algorithm. The performance is also compared with SVM and NN. The results are tabulated in Table 3.4. Clearly SRC performs better than SVM or NN.

Dataset	SRC (%)	SVM (%)	NN (%)
Weizmann	**100**	100	100
HWU	**96**	94	94

Table 3.4: Comparison of the performances on Weizmann and HWU datasets with motion descriptor features.

3.5 Discussion

From the first experiment of evaluating SRC and other classifiers with RP and DS, it can be seen from Table 3.1 and Table 3.2 that, not only the performance with SRC is better but also is more stable in both settings. The performance instability is clearly seen with SVM in both cases. The performance of NN is below both. For the dimensions as low as 70 (comparing to the actual 9600), accuracy with SRC beats the other two classifiers. With feature dimension 70 (less than the number of training samples) the linear system 3.1 is *underdetermined* with dictionary size 70×80 and is solved using l_1 regularization. For feature dimensions greater than 80, the system is *overdetermined* and a unique solution can be obtained in closed form. But, because the solution is expected to select the basis subset to which it belongs to, l_1 norm regularization is still used. To confirm this in the context of video classification as well, a comparison is made on the solutions obtained with l_1, l_2 norm regularization and closed form equation $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$. Figure 3.10 shows the plots of solutions obtained with different regularizations with the linear system 3.1 when it is overdetermined with a feature dimension 128. Clearly sparse solutions with l_1 norm regularization are

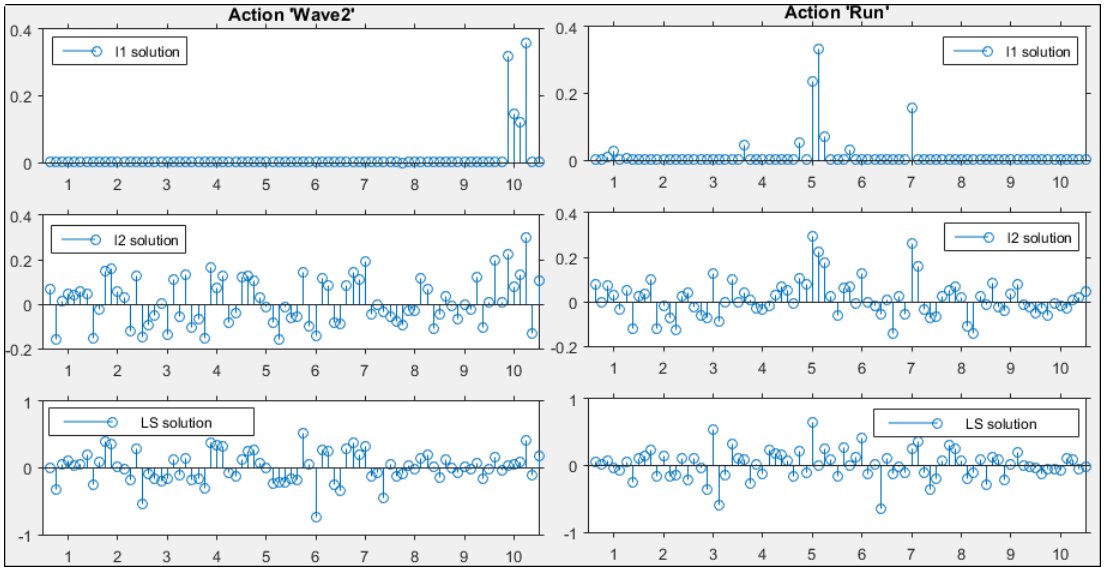

Figure 3.10: Solutions obtained with $\|l\|_1$, $\|l\|_2$ and without regularizing the overdetermined linear system.
The above are plots of solutions of a test sample belonging to class 'Wave2' on left and another in class 'Run' with the GEI features of dimension 128. X-axis denotes the class numbers with 1-bend, 2-jump, 3-jumping in place, 4-jack, 5-run, 6-skip, 7-sidewalk, 8-walk, 9-wave with one hand, 10-wave with two hands. The top row shows sparse solutions obtained by l_1 norm regularization. Middle row shows solution obtained by l_2 regularization and the bottom row is the solution obtained in the closed form.

discriminative by choosing the right subset.

Our second experiment shows that features which capture motion using optical

Approach	Liu et al. [17]	Guo et al. [18]	Guha et al. [19]	Ours with Motion descriptors	Ours with GEIs
Recognition rate (%)	96.77	96.74	98.9	**100**	**98.22**

Table 3.5: Comparison of the proposed approach with others on Weizmann dataset

flow, tend to improve the accuracy of SRC and also SVM and NN. This was expected because optical flow computes intensity gradients of all pixels capturing the finest details of the motion. Third experiment validates the robustness of SRC and other classifiers. While all the three classifiers perform equally well on Weizmann dataset, SRC outdoes on HWU dataset. This implies that SRC is more robust to small misalignments than SVM or NN.
The overall performance of the proposed approach from the above experiments are given in Table 3.6. It is observed that sparse representation based classi-

fication with compressed features performs consistently better than SVM and NN classification methods. We also compare our results with those approaches mentioned in Section 3.2, [17] and [19] [18]. Table 3.5 compares our results with peers. The accuracy of our approach even with simple features such as GEIs - 98.22% - is on par with other works which follow computationally intense feature extraction procedures and or train dictionaries.

Approach	SRC (%)	SVM (%)	NN (%)
RP	**98.22**	95.78	95.11
DS	**96.67**	95.18	94.44
HWU	**96.67**	95.18	94.44
Weizmann with GEIs	**98.22**	95.78	95.11
Weizmann with MD	**100**	**100**	**100**

Table 3.6: Overall performance of SRC
In the above table MD refers to Motion Descriptors.

3.5.1 Technical issues and limitations

The main limitation of our approach is that it requires storage of all data because it is a non-parametric method of classification. The extracted features can be stored instead of raw data. The dimensionality reduction method, RP, is a data independent one and hence features in a lower dimensional subspace can be stored. Another issue to be considered is the testing time. Our approach is slightly slower than SVM as it requires matrix multiplication at every iteration of the algorithms. However, these issues can be lifted as the memory resources and powerful processors are abundantly available now-a-days at reduced costs.

3.6 Conclusion

In this chapter, we proposed a classification strategy for action and gesture recognition based on sparse representation from CS point of view towards the first part of our goal. Experimentally we have shown that the classification can be obtained by computing sparse representation of the test data over a dictionary constructed with compressed features. Our approach bypasses the explicit training phase of

a classifier like SVM and does not train dictionaries before computing sparse representation. This makes it easier to extend or modify the training data. It is also shown that RP is an efficient data-independent dimensionality reduction tool which blends well with SRC. From our experiments, it is also pointed out that exploiting *sparse structures* of the solution, even when the feature dimension is greater than the number of samples results in improved accuracy.

Chapter 4

Joint Classification of Actions using Matrix Completion

In this chapter, we propose to jointly classify actions from more than a single class using Matrix completion. Matrix-completion methods can handle the deficiencies in data very effectively resulting in improved classification accuracy. Features and labels from data are concatenated to form a big matrix with unknown or missing entries in the place of test data labels. Matrix-completion methods fill up these entries using convex rank minimizing tools resulting in classification by predicting test labels. We show that the proposed method achieves comparable performance over the recent works on three action datasets including Weizmann dataset and recently released and more realistic UCF101 dataset.

The work carried out in this chapter was presented at International Conference on Image Processing, Quebec city, 2015 [24].

4.1 Introduction

Another interesting paradigm in the realm of CS, which has gained extensive popularity is the problem of MC [5]. The MC methods are employed in building recommender systems [94] [95]. These systems predict or estimate a user choice based on user history and or other ratings of a product and recommend that product. For example, consider a movie rental company like Netflix [94]. The company often conduct surveys or ask user to rate or review a movie. These ratings will be later used to predict the choice of the user to a new movie and thus either recommend or not recommend that movie. Practically, the rating data will be highly incomplete because not all the users will rate all movies. Figure 4.1 illustrates the rating data from such a company. Estimating these missing values is generally known as MC task. Typically, depending upon the application area, suitable assumptions are made on the underlying data for the MC task. In most of the real world applications, the assumption that the underlying data matrix is a low rank one is exploited [9]. The theory is analogous to sparse representation theory from CS point of view. This chapter presents a novel method to classify actions within a matrix completion framework. We start with theoretical background of matrix completion, links to sparse representation theory and related work in Section 4.2.

4.2 Matrix Completion - MC

The data is acquired from different modalities in many practical scenarios. This data is would be incomplete due to several reasons. The reasons could be anything from improper functioning of sensors to physical contraints like number of sensors or distance between sensors. If the data is from a survey like mentioned above, then user interest in completing the survey would be a major reason. Such scenarios with incomplete data call for MC methods to complete the data matrix assuming some structure in the data. Typically the underlying matrix is a *low rank* one. The popularity of MC is associated to the famous Netflix challenge [9]. The challenge was to come up with a solution to build a *recommender system* from a highly incomplete 100 million user rating data. Figure 4.1 shows an illustration of the user rating matrix.

Low rank matrix approximation [96] of a complete matrix is straight forward using PCA. The given matrix is decomposed using Singular Value Decomposition (SVD) and an approximation is constructed by considering the most influ-

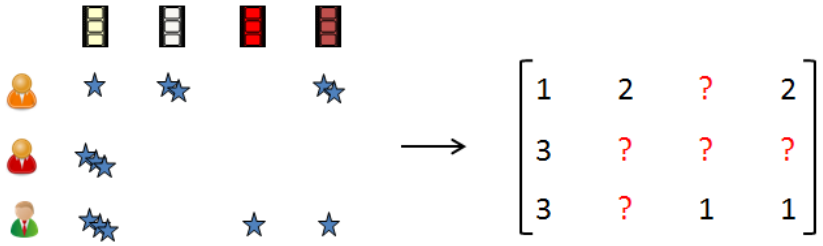

Figure 4.1: Movie rating data

Figure shows a typical user-movie rating data and the implied matrix. The columns are different movies and rows are different user. Each entry is the rating given by a specific user to a particular movie.

ential eigenvalues and eigenvectors. Low rank approximation of an incomplete matrix is much harder to solve. Classical PCA is not applicable for the cases when the data is corrupted or missing. These are posed as *rank minimization* problems. Let $\mathbf{X}$ be an observed matrix with some random entries missing. Let $\mathbf{\Omega}$ be the set of all observed indices ij and $\mathbf{M}_{ij}$ be the values of $\mathbf{X}$ at those ij. The rank minimization problem can be written as Equation 4.1 , from [97].

$$\text{minimize } \text{rank}(\mathbf{X}) \quad \text{subject to } \mathbf{X}_{ij} = \mathbf{M}_{ij} \text{ for } (\text{i,j}) \in \mathbf{\Omega} \tag{4.1}$$

where rank($\mathbf{X}$) is the *rank* of $\mathbf{X}$. Rank of the matrix is the span of rows or columns of a matrix and is given by the number of its non-zero *singular values*. The problem in Equation 4.1 is NP-hard since rank is not a convex function. A close convex heuristic to rank function is the trace or nuclear norm of the matrix [98]. The nuclear norm of a matrix is the sum of its the non-zero singular values. Thus the convex relaxation of 4.1 is Equation 4.2.

$$\text{minimize } \|\mathbf{X}\|_* \quad \text{subject to } \mathbf{X}_{ij} = \mathbf{M}_{ij} \text{ for } (\text{i,j}) \in \mathbf{\Omega} \tag{4.2}$$

where $\|\mathbf{X}\|_* = \sum_{i=1}^{n} \sigma_i(\mathbf{X})$ is the nuclear norm of $\mathbf{X}$ and σ_i is it's i^{th} singular value. Equation 4.2 can be solved efficiently by semi-definite programming.

4.2.1 Connections to Sparse Representations

Some analogies between sparse representations and MC can be observed [99].

1. Both methods seek compact solutions in respective domains.

2. Rank of the matrix is the count of non-zero singular values which is similar to the definition of $\|l\|_0$ norm.

Figure 4.2: Action classification with Matrix Completion
Proposed framework is shown above. Features and labels from training data are concatenated and arranged class-wise as shown. Test data features and labels are appended with zeroes in place of test labels. Using rank minimization techniques, the test labels are predicted.

3. Nuclear norm is the tightest convex relaxation of rank which is comparable to that of $\|l\|_1$ norm to $\|l\|_0$.

4. The random observed entries $\mathbf{X}$ are analogous to the random measurements of the dictionary in the CS theory.

The underpinning theories of CS like RIP and incoherence are generalised to MC in [97]. Applications include *collaborative filtering* which is to predict users' interest based on reviews from other users, extensively used with the companies promoting any new releases or products [100]. *System Identification* is another application in control theory where the problem is to predict the system state at a given time based on the input and output values [101]. Estimating the sensor node positions in a wireless sensor network is another application of MC where the node location indicates the local area which it is monitoring [102]. The data available would be distances and angles between the neighbouring nodes only. Commonly used algorithms to solve the problems of Equation 4.2 are *OptSpace* [103] and Singular Value Thresholding (SVT) [104]. Transduction classification with matrix completion was introduced by Goldberg et al in [105]. Based on this work these methods were used for *multi-label* classification tasks. In [20] and [106], matrix completion is used for multi-label image classification. This is extended to handle images with multiple viewpoints in [107]. Also relation extraction problem of natural texts in [108] is solved using matrix completion.

4.3 Joint classification of actions

General classification methodologies require the classifying algorithm to run as many times as the number of test samples. In this work, we propose to jointly classify actions from multiple test videos in a matrix completion setting in feature and label space. This work adapts the framework introduced in [105] and [20]

but in a different context. We focus on *multi-class classification* of actions from videos rather than *multi-label classification* of images. A matrix is constructed by concatenating labels and features of training data and appended with unknown labels and features from test data. Completing this matrix which is void at test label indices, results in classification. This is transduction classification, since the statistics of the test data are also included in completing the matrix. Figure 4.2 illustrates our proposal of applying MC for action classification. We now give the theory behind this approach.

4.3.1 Theoretical Framework

Consider that we have some training data comprising of feature vectors $\mathbf{x}$ and corresponding labels $\mathbf{y}$. Let the feature vector of a training sample be in $\mathbf{R}^m$ and N_{tr} be number of training samples from all the classes. Let $\mathbf{X}_{train} \in \mathbf{R}^{m \times N_{tr}}$ be the matrix with feature vectors of training samples from all classes as columns. In a similar fashion, let $\mathbf{X}_{test} \in \mathbf{R}^{m \times N_{test}}$ be the matrix formed with N_{test} test samples. Conventionally, the classification problem can be expressed as $\mathbf{y} = \mathbf{f}(\mathbf{x})$, where $\mathbf{f}$ is the mapping from input feature space $\mathbf{x}$ to the output label space $\mathbf{y}$. The mapping could be a linear one or a non-linear one. A linear classifier finds a decision boundary defined by the parameters $[\mathbf{W}^T, \mathbf{b}]$ such that $\mathbf{y}_i = \mathbf{W}^T\mathbf{x}_i + \mathbf{b}$ where $(\mathbf{x}_i, \mathbf{y}_i)$ is the feature and label pair from the training data for $i = 1, 2, \ldots N_{tr}$. This parameter set is then used for classifying test data. Exploiting this linear dependence of labels on features, the matrix formed by the concatenation of labels and features will be of low rank. For this reason, MC approach for classification can be considered as a linear classification problem. Let $\mathbf{Y}_{train} \in \mathbf{R}^{n \times N_{tr}}$ and $\mathbf{Y}_{test} \in \mathbf{R}^{n \times N_{test}}$ be the label matrices from train and test samples in n classes. Arranging these matrices as shown in Equation 4.3, will result in a matrix, say $\mathbf{A}$, of low rank.

$$\mathbf{A} = \begin{bmatrix} \mathbf{Y}_{train} & \mathbf{Y}_{test} \\ \mathbf{X}_{train} & \mathbf{X}_{test} \end{bmatrix} \Rightarrow \begin{bmatrix} \mathbf{Y}_{train} & ? \\ \mathbf{X}_{train} & \mathbf{X}_{test} \end{bmatrix} \tag{4.3}$$

Bias $\mathbf{b}$ of the linear model can be handled by appending a row of $\mathbf{1}s$ to $\mathbf{A}$. $\mathbf{Y}_{test}$ is unknown and has to be predicted from $\mathbf{A}$ by optimizing the following problem. Denoting $\mathbf{A_X} = [\mathbf{X}_{train}\ \mathbf{X}_{test}]$, $\mathbf{A_Y} = [\mathbf{Y}_{train}\ ?]$ and $\mathbf{A_1} = \mathbf{1}^{\mathrm{T}}$ as the observed submatrices and let $\mathbf{A_0} = [\mathbf{A_{0Y}}; \mathbf{A_{0X}}; \mathbf{A_1}]$ be the underlying low rank

completed matrix, the problem can be stated as

$$\begin{aligned} &\text{minimize} \quad \text{rank } (\mathbf{A}) \\ &\text{subject to} \quad \mathbf{A_0} = \mathbf{A} = [\mathbf{A_Y}; \mathbf{A_X}; \mathbf{A_1}] \end{aligned} \tag{4.4}$$

It is to be noted that the constraint in Equation 4.4 is with respect to the observed entries. Equation 4.4 is hard to solve due to non-convex *rank* function. The convex alternative to Equation 4.4 can be obtained by replacing rank with *nuclear norm* function. To account for errors in features and labels, two loss terms are introduced in [105] and [20]. Least squares loss term is used for feature loss and log loss for label errors. A non-negativity constraint is also imposed on the feature submatrix. Putting all together we arrive at the final formulation of the problem.

$$\begin{aligned} &\text{minimize} \quad \mu\|\mathbf{A}\|_* + l_X(\mathbf{A_X}) + \lambda\, l_Y(\mathbf{A_Y}) \\ &\text{subject to} \quad \mathbf{A_X} \geq \mathbf{0} \;\text{ and }\; \mathbf{A_1} = \mathbf{1}^{\mathrm{T}} \end{aligned} \tag{4.5}$$

where

$$\begin{aligned} l_X(\mathbf{A_X}) &= \sum_{ij \in \mathbf{A_X}} (\mathrm{a}_{ij} - \mathrm{a}_{0ij})^2 \\ l_Y(\mathbf{A_Y}) &= \sum_{ij \in \mathbf{Y}_{train}} \frac{1}{\gamma} \log(1 + \exp(-\gamma \mathrm{a}_{ij} \mathrm{a}_{0ij})) \end{aligned} \tag{4.6}$$

The parameters μ and λ are positive weights for better adaptation of features and labels and γ is a regularizer used to smooth the log loss.

4.3.2 Algorithm

The problem in Equation 4.5 is convex but not smooth. Typically the feature matrices in computer vision applications are large and using conventional approaches like interior-point methods cannot handle such large-scale data. An alternative approach is Fixed Point Continuation (FPC) method from [109]. Fixed point methods consist of two alternating steps of gradient computation and shrinkage operation. Further to speed-up the algorithm, these two steps are repeated until a specified error tolerance for a sequence of fixed values of a parameter (μ, nuclear norm weight in this case). For every parameter value, the initial point is the final value obtained from the previous parameter value. The gradient computation of the matrix is computed in two steps. Since the loss on features and labels are modelled using different functions, the gradients for each part, feature part and label part, are computed independently and then combined to get the gradient

of the complete matrix. The gradients are given by Equation 4.7

$$
\begin{aligned}
g_{Yij} &= \frac{\lambda}{\mid \Omega_Y \mid} \frac{-a_{ij}}{1+\exp(\gamma \mathrm{a}_{ij}\mathrm{a}_{0ij})} \text{ for } ij \in \mathbf{A}_{\mathbf{Y}_{train}} \\
g_{Xij} &= \frac{1}{\mid \Omega_X \mid}(\mathrm{a}_{ij} - \mathrm{a}_{0ij}) \text{ for } ij \in \mathbf{A}_{\mathbf{X}}
\end{aligned}
\tag{4.7}
$$

where $\mid \Omega_Y \mid$ denotes number of entries in observed label part $\mathbf{A}_{\mathbf{Y}_{train}}$ and $\mid \Omega_X \mid$ denotes the number of elements in feature part $\mathbf{A}_{\mathbf{X}}$. After computing

Algorithm 3: Classification with Matrix Completion adapted from [20]

Input : Incomplete Initial Matrix: $\mathbf{A}$
Parameters : λ , γ , set of $\mu = \{\mu_1 > \mu_2 \, . \, . \, > \mu_{\mathrm{k}}\}$
error tolerance : ϵ

Output: Predicted labels $\mathbf{Y}_{test}$

Initialize:

1. Initial labels: $\mathbf{Y}_{test} = \mathbf{0}$
2. $\mathbf{A_0}$ as rank-1 approximation of $\mathbf{A}$

For each μ **do**
  while relerr $> \epsilon$ **do**
    Compute $\mathbf{g_X}$ and $\mathbf{g_Y}$ using Equation 4.7
    $\mathbf{g_A} = [\mathbf{g_Y}; \mathbf{g_X}]$
    Gradient step : $\mathbf{D} = \mathbf{A_0} - \tau \mathbf{g_A}$
    $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V^T}] = SVD(\mathbf{D})$
    Shrink : $\mathbf{A_0} = \mathbf{U} S_{\tau\mu}(\mathbf{\Sigma}) \mathbf{V^T}$
    Project : $\mathbf{A_{0X}} = \mathbf{max}(\mathbf{A_{0X}}, \mathbf{0})$
    Project : $\mathbf{A_{01}} = \mathbf{1^T}$
    Current objective : $\mathbf{C}^{\mathrm{i}}$ using Equation 4.5
    relerr $= \mathbf{C}^{\mathrm{i}} - \mathbf{C}^{\mathrm{i-1}}$
  end while
end For

$\mathbf{Y}_{test} = \mathbf{max}(\mathbf{A_{0Y}}_{test})$

the gradients, nuclear norm is minimized using shrinkage operation. Shrinkage operation is to apply some threshold on the obtained singular values of the SVD of observed matrix. This was introduced in [104] as SVT algorithm for MC. The shrinkage operation denoted by $S_{\tau\mu}$ is function of gradient descent step size τ and continuation parameter μ such that $S_{\tau\mu}(\sigma) = max(0, \sigma - \tau\mu)$. To satisfy the constraints in 4.5, *'max'* is used to project the matrix onto non-negative orthant and the bias is handled by projecting the matrix to the feasible region. Generally the values of μ are set from a highest value and decrease upto a predefined lowest value, (of the order of $1e-5$), using a reduction parameter η_μ. The largest value is set as $\mu_1 = \sigma_1 \eta_\mu$, where σ_1 is the largest singular value of the initial observed

Figure 4.3: A snapshot of KTH dataset

matrix. The subsequent μ values are given by $\mu_2 = \mu_1\eta_\mu$, $\mu_3 = \mu_2\eta_\mu$ and so on. The complete algorithm is detailed as Algorithm 3.

4.4 Experiments

We evaluate our approach on three datasets of varying complexities. (1) Weizmann dataset (2) KTH dataset (3) UCF101 dataset. The details of Weizmann dataset are given in Chapter 3, Section 3.4.1. The details of the other two datasets follow next.

4.4.1 KTH dataset

KTH dataset is released by Schuldt et al. [110] and is the earliest databases with large number of examples in each class. Figure 4.3 shows the snapshot of this dataset. It is comprised of videos from 6 action classes - boxing, handclapping, handwaving, jogging, running, walking - recorded in 4 scenarios - s1, s2, s3, s4. These scenarios are outdoors, outdoors with scale variation, outdoors with different clothing and indoors respectively. There are 25 persons performing the actions under each setting. Approximately there are 600 videos in total and each video has multiple sequences of the same action. If the video is split according to sequences, then there will be 2391 sequences with around 400 sequences in each class.

4.4.2 UCF101 dataset

UCF101 dataset is the largest video dataset released recently by Soomro et al. in [111]. There are 101 action classes divided into five categories - Human-human interaction, human-object interaction, sports, body motion only, playing musical

Figure 4.4: UCF101 dataset

A subset of UCF101 dataset showing some of the classes. The five action categories are indicated with different colour outlines. This dataset captures a wide range of actions in realistic environments.

instruments. A snapshot of this dataset is shown in Figure 4.4. The five categories are differentiated with the colour-coding in the figure. Each class consists of approximately 100 video clips, divided into 25 groups. Each group has 4 - 7 clips and these clips share some features like same actor or similar background etc. There are three non-overlapping splits (train and test) specified for this dataset.

4.4.3 Evaluation

Since each dataset is of varied complexity, different types of features are extracted from each dataset. Binary vectors in $\{0, 1\}$ are the labels of dimension depending on the number of classes in the dataset. In all the experiments, test labels are set to zero initially.

Weizmann dataset

For this dataset, we use the previously extracted GEIs as features, with each GEI of dimension 9600×1. Labels are binary of dimension 10×1 corresponding to 10 classes. With LOOCV, one subject from all classes will be for testing and remaining will form the training set. For each subject, the feature matrices $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$ are of size $n \times 80$ and $n \times 10$ respectively. The label matrices $\mathbf{Y}_{train}$ and $\mathbf{Y}_{test}$ are of size 10×80 and 10×10. Having witnessed the efficiency of RP as dimensionality reduction tool in Chapter 3, Section 3.4, the same is used here to reduce the feature dimensions. Each GEI is reduced to 1024, 2048 and 6144. Experiments are performed with these random projected features. In each case, the corresponding size of $\mathbf{A} = [\mathbf{A_Y}; \mathbf{A_X}; \mathbf{1}^T]$ would be 1035×90, 2059×90 and 6155×90. Initial $\mathbf{Y}_{test}$ values are considered zeroes for all cases. The parameters are set to $\lambda = 0.1$, $\gamma = 1$, reduction parameter to compute set of μ values, η_μ

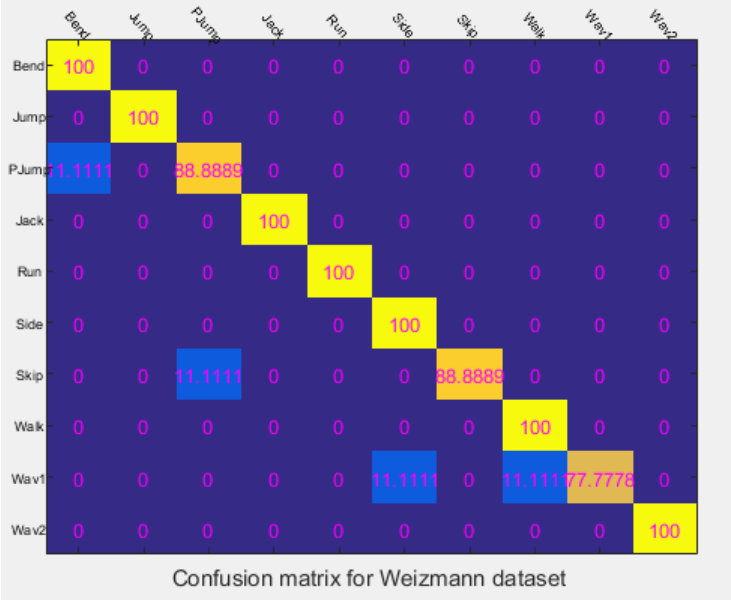

Figure 4.5: Confusion matrix for Weizmann dataset for 6144 feature dimension.

is set to 0.25 and the final least value of μ is set to $1e-8$. Stopping criteria of the inner while loop in the algorithm is taken as the difference between the current and previous objective values to be less than $\epsilon = 1e-2$. The results are tabulated in Table 4.1. The confusion matrix for the best case, with projected dimension 6144 is shown in Figure 4.5.

True dimension	Reduced Dimension	Recognition rate(%)
9600	1024	91.11
9600	2048	94.44
9600	6144	**95.55**

Table 4.1: Weizmann dataset results with MC

KTH Dataset

For this dataset, dense trajectory based descriptors [47] explained in Chapter 2, Section 2.1.4, are extracted. For each video, trajectories are tracked based on optical flow at densely sampled pixels. A spatio-temporal volume of size $32 \times 32 \times 15$ surrounding the trajectory is considered to compute the descriptors. This volume is subdivided into small spatio-temporal patches of size $2 \times 2 \times 3$. For each patch, HOG (explained in Chapter 2, Section2.1.2) , HOF, MBHx, MBHy (explained in Chapter 2, Section 2.1.2), and Trajectory (TR) descriptors are computed. For HOG, MBHx and MBHy the directions are quantized into 8 bins with the resulting size of each descriptor being 96. For HOF, the number of bins is 9 including

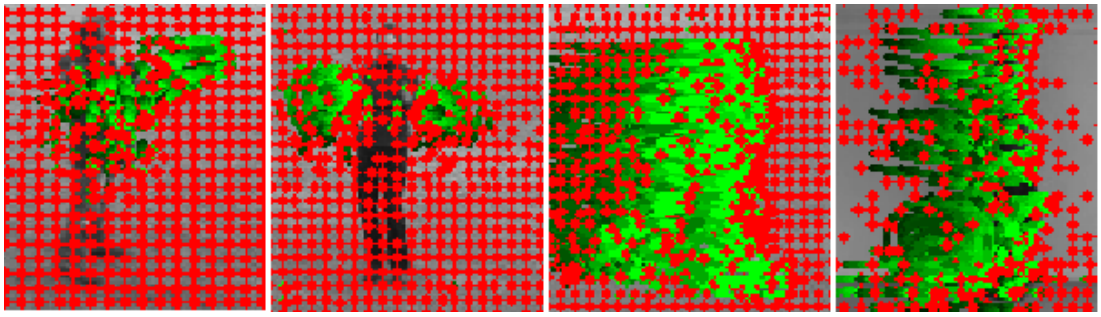

Figure 4.6: Dense trajectories on KTH dataset
The above figure shows the dense trajectories(in green colour) tracked for actions boxing, handwaving, running and walking respectively. In red colour are the dense points sampled but will not be tracked.

the zero bin resulting in a descriptor of size 108. The TR descriptor is computed by the displacements $(\triangle\ x, \triangle\ y)$ from frame to frame. Each trajectory extends for 15 frames giving a descriptor of size 30. These are first reduced to half of actual size using PCA. The reason to use PCA here is to extract same features (as in [47]) for comparison of approaches. Application of PCA here can thus be treated as one of the steps in feature extraction process. RP is applied after the final feature descriptor is obtained. Figure 4.6 shows the trajectories for four actions boxing, handwaving, running and walking of the dataset. Before going further we briefly give the details of fitting a Gaussian Mixture Model to the set of data points next.

Gaussian Mixture Model - GMM

A GMM is obtained by a clustering algorithm for unsupervised learning of data similar to k-means algorithm. k-means is based on 'hard'assignment of data to cluster centers (means) using the Euclidean distance metric [49]. This metric is not a proper choice when the data clusters are expected to have considerable covariance. Following the description and notation in [112], given a dataset of m training points $\{\mathbf{x}_1, \mathbf{x}_2, \ \ldots\ \mathbf{x}_m\}$, GMM tries to estimate a joint distribution of data such that $p(\mathbf{x}_i, \mathbf{z}_i) = p(\mathbf{x}_i|\mathbf{z}_i)p(\mathbf{z}_i)$. The variables $\mathbf{z}_i$ are drawn from a multivariate Gaussian parametrized by $\{\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. Let j in $\{1, \ \ldots\ , k\}$ are the values $\mathbf{z}_i$ can take and $\mathbf{x}_i|\mathbf{z}_i = j \sim \mathcal{N}(\mu^j, \Sigma^j)$. $\boldsymbol{\phi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the priors (where $\boldsymbol{\phi}^j$ gives $p(\mathbf{z}_i = j)$), means and covariances of each Gaussian respectively. These parameters are estimated using the likelihood of the data given by

$$l(\boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{m} \log\ p(\mathbf{x}_i; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{4.8}$$

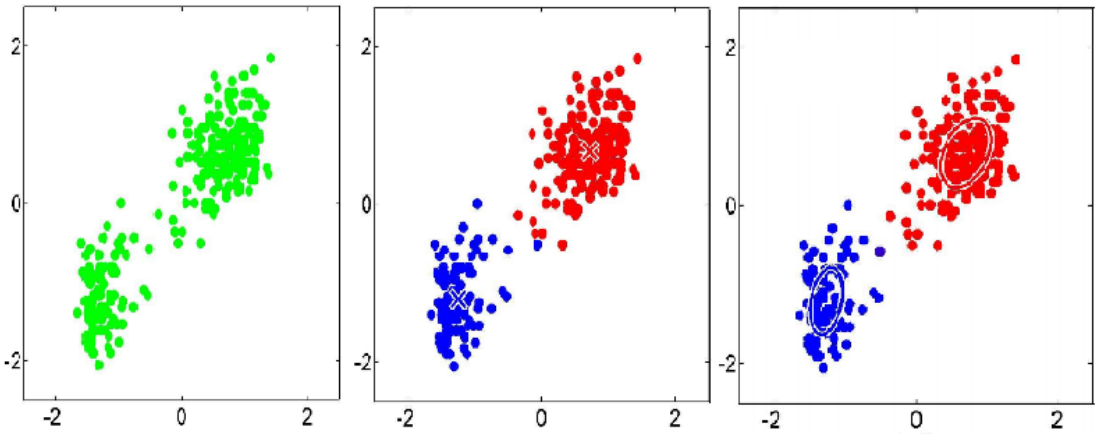

Figure 4.7: Clustering with k-means and GMM

The above figure shows the clustering methods k-means and fitting a GMM. The original 2D data is to the left, data clustered with k-means in the middle learns cluster centers iteratively and data clustered with a GMM on the right. A mixture of Gaussians are fit to the data by learning means and variances within each cluster. Figure adapted from [113]

Each data point $\mathbf{x}_i$ is assigned to one of the k Gaussians using $\mathbf{z}_i$. The values are $\mathbf{z}_i$ are unknown and hence the parameters are estimated iteratively using EM algorithm. The EM algorithm basically consists of E-step (Expectation) where initially $\mathbf{z}_i$ are randomly guessed. This is followed by and M-Step (Maximization) updates the parameters using the $\mathbf{z}_i$ from E-step. These two steps are repeated until convergence. The E-step and M-step are given by the following equations.

E-Step:

$$\mathbf{w}_i^j = p(\mathbf{z}_i = j \mid \mathbf{x}_i; \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{4.9}$$

M-Step:

$$\begin{aligned} \boldsymbol{\phi}^j &= \frac{1}{m}\sum_{i=1}^{m} \mathbf{w}_i^j \\ \boldsymbol{\mu}^j &= \frac{\sum_{i=1}^{m} \mathbf{w}_i^j \mathbf{x}_i}{\sum_{i=1}^{m} \mathbf{w}_i^j} \\ \boldsymbol{\Sigma}^j &= \frac{\sum_{i=1}^{m} \mathbf{w}_i^j (\mathbf{x}_i - \boldsymbol{\mu}^j)(\mathbf{x}_i - \boldsymbol{\mu}^j)^T}{\sum_{i=1}^{m} \mathbf{w}_i^j} \end{aligned} \tag{4.10}$$

In our experiments we set $k = 256$. For each descriptor, $256,000$ PCA reduced features are randomly sampled from all the classes in the training data. A GMM for each case is learnt with these points using Matlab toolbox vlfeat from [114].

Fisher Vector - FV encoding

Using the GMMs from above, each descriptor is encoded as FV following [115]. Brief introduction on this encoding was given in Chapter 2, Section 2.1.5. Fisher encoding captures the first and second order statistics between the data and the mixture model. Using the same notation as above, for each $\mathbf{x}_i$, the posterior probability [116] is given by

$$\mathbf{p}_{ik} = \frac{exp[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}^k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}^k)]}{\sum_{j=1}^{k} exp[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}^k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}^k)]} \tag{4.11}$$

For each point $\mathbf{x}_i$ in the training set, the mean and covariance discrepancies $(\mathbf{u}_{dk}, \mathbf{v}_{dk})$ are evaluated by the following equations.

$$\begin{aligned} \mathbf{u}_{dk} &= \frac{1}{m\sqrt{\boldsymbol{\phi}^k}} \sum_{i=1}^{m} \mathbf{p}_{ik} \left(\frac{\mathbf{x}_{di} - \boldsymbol{\mu}_{dk}}{\boldsymbol{\sigma}_{dk}} \right) \\ \mathbf{v}_{dk} &= \frac{1}{m\sqrt{2\boldsymbol{\phi}^k}} \sum_{i=1}^{m} \mathbf{p}_{ik} \left[\left(\frac{\mathbf{x}_{di} - \boldsymbol{\mu}_{dk}}{\boldsymbol{\sigma}_{dk}} \right)^2 - 1 \right] \end{aligned} \tag{4.12}$$

d takes the values $1, 2, \ldots, D$ where D is the dimension of the descriptor after PCA. The final fisher representation $\mathcal{F}(\mathbf{X})$ of the training set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_m\}$ is given by the concatenation of all $(\mathbf{u}_k, \mathbf{v}_k)$ as follows.

$$\mathcal{F}(\mathbf{X}) = \begin{bmatrix} \mathbf{u}_1 \\ \cdot\cdot \\ \mathbf{u}_k \\ \mathbf{v}_1 \\ \cdot\cdot \\ \mathbf{v}_k \end{bmatrix} \tag{4.13}$$

The dimension of fisher encoded feature descriptor is thus $2Dk$. In our experiments we compute the improved fisher encodings for each descriptor with corresponding GMM using Matlab toolbox vlfeat from [114]. The improved version [53] applies Hellinger's kernel to each dimension of fisher vector to get the square-rooted vector and then normalized using $\|l\|_2$ norm. The combined encoded feature vector is obtained by concatenating encoded HOG, HOF, MBHx, MBHy and TR descriptors. The dimension of the combined descriptor is 109056. To avoid memory issues with Matlab in computation of SVD required by MC algorithm 3, we resort to RP to reduce the dimension. These random projected Fisher vectors are used in our experiments.

The dataset is divided into training set of 16 persons and remaining 9 persons

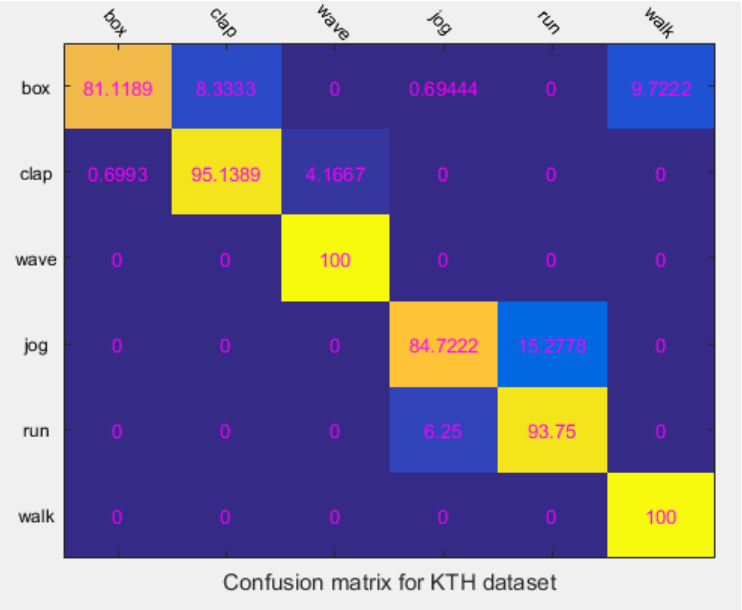

Figure 4.8: Confusion matrix for KTH dataset for 6144 feature dimension.

for testing as recommended in [110]. This division leads to approximately 250 videos in each training class and 144 test samples in each class. $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$ are of size $n \times 1524$ and $n \times 144$ respectively where n is the dimension after RP. $\mathbf{Y}_{train}$ and $\mathbf{Y}_{test}$ are of size 6×1524 and 6×144. The parameters are set to $\lambda = 0.01$, $\gamma = 30$, reduction parameter to compute set of μ values, η_μ is set to 0.25 and the final least value of μ is set to $1e{-}8$. The stopping criteria is chosen as in the experiment with Weizmann dataset. Table 4.2 shows the recognition rates obtained with MC algorithm for different dimensions of the feature descriptors. The confusion matrix for the best case with projected feature dimension 6144 is shown in Figure 4.8.

True dimension	Reduced Dimension	Recognition rate(%)
109056	1024	87.11
109056	2048	85.83
109056	3072	88.04
109056	6144	**92.92**

Table 4.2: Recognition rates on KTH dataset with MC

UCF101

We use the dense trajectory based descriptors from [117] for this dataset. The snapshot of trajectories on 4 classes - Apply Eye Makeup, Cricket shot, Fencing and Hulahoop - is shown in Figure 4.9. For this dataset, we initially perform

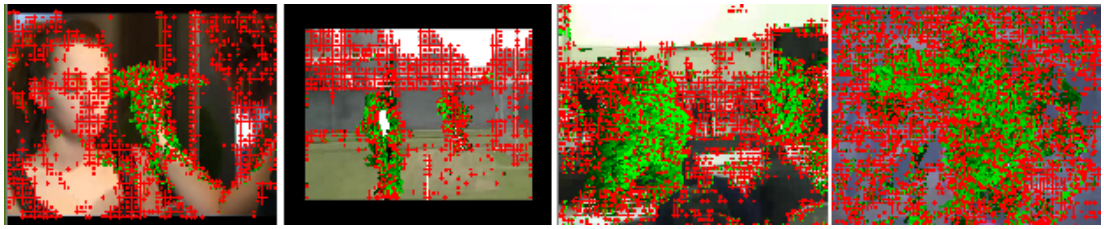

Figure 4.9: Dense Trajectories on UCF101 dataset

Figure shows the tracking of dense trajectories on four of 101 classes. The classes shown above from left are (1) Apply Eye Makeup (2)Cricket shot (3) Fencing (4)Hula hoop

experiments using the features available from [117]. These are BoVW encoded (please refer to Chapter 2, Section 2.1.5 for the details of this encoding) features. The feature descriptors evaluated over the trajectory are HOG, HOF and MBH. From each descriptor type, $100,000$ data points are sampled randomly from the training data. Using k-means algorithm [49], a 4000 word codebook is learnt for each descriptor. The data points are assigned to a codeword based on the Euclidean distance metric. The representation of each video is thus in terms of histograms of these codewords for each descriptor. The 25 groups in the dataset are divided into 3 non-overlapping sets of 18 groups of training data and remaining 7 groups of test data. Considering 4 clips in each group of training data we have $\mathbf{X}_{train}$ of size $n \times 7272$. Size of $\mathbf{X}_{test}$ varies according to number of clips in each test groups. $\mathbf{Y}_{train}$ and $\mathbf{Y}_{test}$ are of size 101×7272 and $101 \times N_{tst}$ where N_{tst} is the number of test samples in a class. The dimension of the concatenated descriptor after encoding would be 8000 with HOG(4000), HOF(4000) and MBH(4000). Again using RP these are reduced to $n = 1000, 2000$ and 4000. The recognition rates obtained using MC method for the three splits for chosen projected dimensions are shown in Table 4.3. The best recognition rate **52.06%** is obtained at a dimension 4000.

Reduced Dimension	Split # 1	Split # 2	Split # 3	Average(%)
1000	45.59	46.87	48.54	47
2000	48.68	50.88	50.19	49.92
4000	51.09	52.07	53.03	**52.06**

Table 4.3: Performance of MC on UCF101 dataset - 1

The feature descriptors are formed by concatenation of BoVW encoded HOG, HOF and MBH descriptors. Split# 1 represents the first, Split# 2 the second and Split# 3 the third split of the dataset. The recognition rate for each split and average are given for each dimension. It is seen that the increasing the dimension improves the recognition rate.

The BoVW encoded features are sparse due to the 'hard'assignments of the data points to the codewords and hence may not encapsulate the statistics of the dataset. Therefore the experiments are repeated with FV encoding as described in Chapter 2, Section 4.4.3. Since there are 3 splits of the dataset, GMMs are learnt independently for each train and test split. We randomly sample 256000 PCA reduced descriptors from the training set and k is set to 256 for the number of Gaussian modes. FV encoding is followed on the reduced descriptors using the corresponding GMMs. Like KTH dataset, the concatenated encoded descriptor dimension in this case also is 105096. These features are further reduced by RP to $1024, 2048, 4096, 6144$. Feature and label sub-matrices are formed in the same way as with the BoVW encoded features and classification with MC algorithm is obtained. The results are tabulated in Table 4.4 for different reduced dimensions. It is noted that Fisher vector encoding increases the best performance with BoVW by 17%. The best recognition rates **70.10%** and **70.06%** were obtained on the second split with 4096 dimension and on the third split with 6144 dimension respectively.

Reduced Dimension	Split # 1	Split # 2	Split # 3	Average(%)
1024	59.26	60.71	59.78	59.91
2048	62.98	64.93	65.48	64.46
4096	67.71	70.10	69.37	69.06
6144	68.67	69.38	70.06	**69.37**

Table 4.4: Performance of MC on UCF101 dataset - 2
The feature descriptors are formed by concatenation of FV encoded HOG, HOF, MBH and TR descriptors. Comparing with the results in Table 4.3, we note that Fisher vector encoding improves the best performance by 17%.

The confusion matrices are also plotted for the best recognition rates obtained with Fisher encoded features. Figure 4.10 shows the confusion matrix for Split # 2 with 4096 dimension and Figure 4.11 shows the confusion matrix for Split # 3 with 6144 dimension.

4.4.4 Discussion

Supervised classifiers assign a label to a single test sample at a time after being trained on labelled test data. The proposed classification method is used to classify multiple test samples as a single matrix completion problem. We now analyse our results tabulated in Table 4.1, Table 4.2, Table 4.3 and Table 4.4.

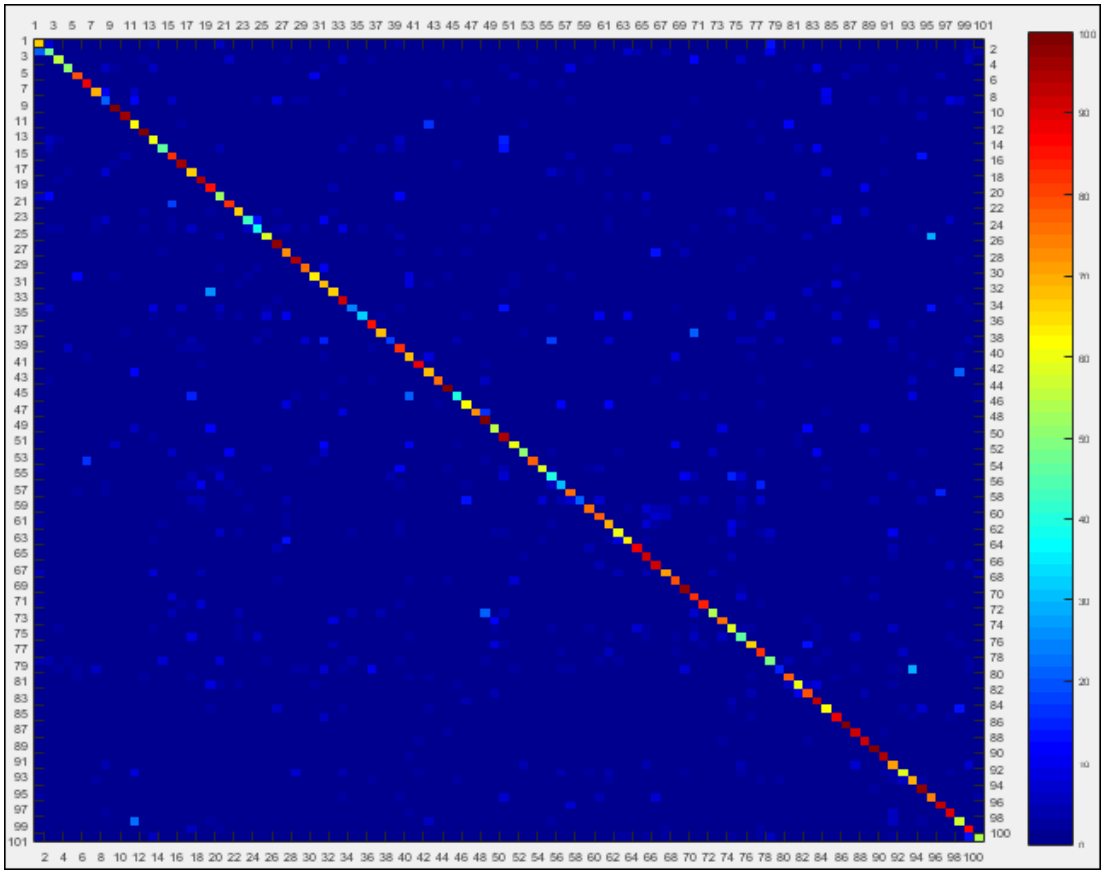

Figure 4.10: Confusion matrix on UCF101 dataset on Split # 2

Weizmann dataset has been extensively used for testing the performance of any new classification strategy, neverthless the testing would be with respect to a single sample. The highest recognition rate obtained with MC method is **95.55%**, which is comparable to many state of the art methods including our work in previous chapter [23]. For this dataset the test sub-matrix consists of samples from all classes proving the ability of MC method in classifying actions from different classes.

The best recognition rate on KTH dataset in **92.92%**. This is very close to 94.2% from [47] and 94.53% from [118]. The recognition rate obtained with linear SVM [92] with the reduced features of same dimension (6144) was **95.72%**

For the UCF101 dataset, the best average recognition rate is **69.37%** with FV encoding. The confusion matrices shown in Figure 4.10 and Figure 4.11 are sparse and nearly diagonal which reflects the ability of MC in discriminating between the classes. Considering the reduced dimension of the feature descriptor in our experiments, the performance on this dataset is much better when compared to the baseline of 43.9% of [111] and 63.3% of [119]. This is less than recogni-

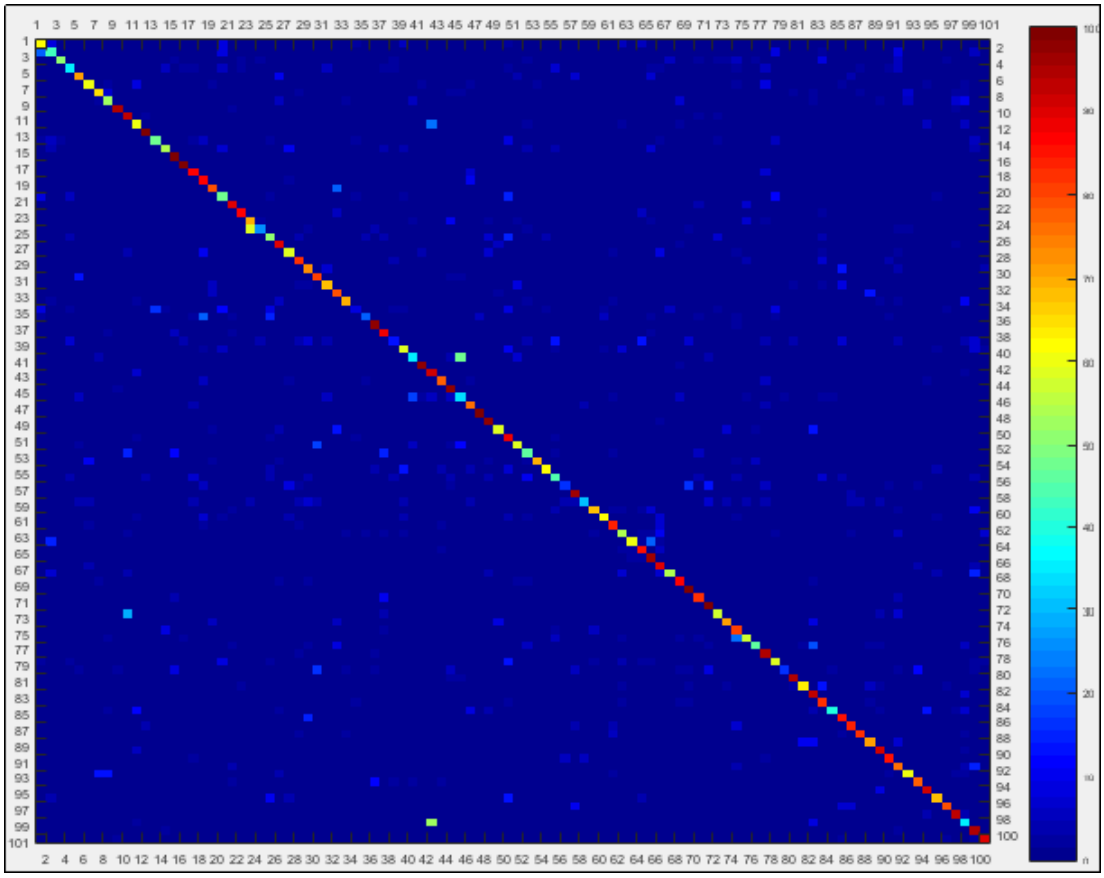

Figure 4.11: Confusion matrix on UCF101 dataset on Split # 3

The confusion matrices shown above and in Figure 4.10 are sparse and diagonal implying the ability of the MC method to distinguish between the classes.

tion rate of 84.8% from [120] in which linear SVM is used for classification with Fisher encoded features. As observed from the Table 4.4, the performance could be expected to improve with increase in the random projected dimension. The main assumption for the applicability of MC method, the underlying low rank structure, is weak due to lot of intra-class variation as seen in Figure 4.12. It is probably due to this intra-class variation, there are some outliers observed from the confusion matrix in Figure 4.11. The class 'High jump'indexed '40 'has a main outlier in class 'Javelin throw 'indexed '45 '. Similarly 'Horse riding'indexed '42 'has a main outlier in class 'Walking with dog 'indexed '98 '.

Technical issues and limitations

Our MC approach of classification is a non-parametric approach, and hence all the training data features have to be stored unlike the case with any conventional classifier which learns a set of parameters from the data. With the availability of

Figure 4.12: Sample frames from UCF101 dataset
Figure shows random frames from two action classes in the UCF101 dataset. Top row is class 'Biking' and bottom row is class 'Archery'. It is evident that the intra-class variation is quite significant.

memory resources at reduced costs these days, the memory requirements to store the data is not a major limitation. Moreover, the random projected features could be stored rather than the raw data which require significantly less memory than the raw data. Computationally, the main burden on the algorithm is the SVD of the data matrix in every iteration. Developing algorithms which can speed up this computation could be considered as one of the extensions to this work.

4.5 Conclusion

In this chapter, we have proposed a novel method for joint classification of actions from multiple videos using MC, which relies on the *low rank structure* of the feature-label matrix. In this work, the unknown labels of test data are the only missing entries, making the classification method a supervised one with transduction. The approach is evaluated on three datasets of varied complexity. The performance on Weizmann and KTH datasets is on par with the state of the art and quite promising on UCF101 data. From the experiments, it can be deducted that the low rankness of the feature-label matrix depends not only the nature of the dataset which is intuitive, but also on the features. The experiments on the UCF101 dataset with BoVW and FV encoded features show that this measure (low rankness) can be varied with different features. Also it would be interesting to see if the matrix completion method can be extended to the cases when some features are missing besides the test labels. In the next chapter, we extend this classification method to handle feature deficiencies and explore in detail about the low rank measure on two different features.

Chapter 5

Deep Action Classification via Matrix Completion

In previous chapter we proposed matrix completion method to classify actions in multiple test videos simultaneously. A matrix is constructed out of features and labels from training and test videos. Since the labels of test data are unknown, the matrix would be incomplete in the test labels. Minimizing the rank of this matrix would result in estimating the unknown test labels and thus classification. In this chapter, we extend the matrix completion framework to deal with the deficiencies in the features as well, besides unknown test labels using deep features. Deep learning has evolved as an efficient tool for feature extraction in many large-scale image based applications. Exploiting the techniques from both domains, we propose a novel solution to the problem of simultaneous classification of actions from multiple test videos with deep features using matrix completion methods. Learned features from a convolutional neural network and corresponding labels from data are concatenated to form a big matrix with unknown or missing entries in the place of test data labels. Convex rank minimization algorithms are used to complete this matrix. The proposed method achieves stable performance even in situations with more than 50% of features and labels jointly missing.

The work carried out in this chapter is presented 24^{th} European Signal Processing Conference - EUSIPCO 2016.

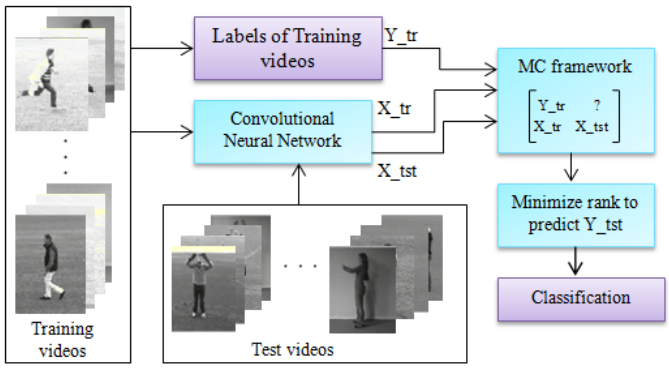

Figure 5.1: Matrix completion with deep features

Proposed approach : A convolutional neural network is trained to learn class specific features from given training videos. The trained network is used to extract features X_tr and X_tst from train and test videos. Label vectors Y_tr corresponding to the training videos are generated. Features and labels are arranged as shown in MC framework with zeroes replaced in the place of unknown entries. Classification is achieved by completing the matrix using rank minimization techniques.

5.1 Introduction

Real world data is rarely perfect or clean. This can be due to human error, equipment failure, system generated error etc. For example surveillance videos are of low resolution often, also the relevant data may be occluded many times. This results in incomplete data leading to missing features. It is required that a classifier be robust to missing features. In supervised learning another issue is the availability of labelled data. Labelling is manual and laborious for large datasets. With only some of the data labelled, the problem would be semi-supervised learning, where a classifier is trained on labelled and unlabelled training data. We therefore seek a classifier which is robust to deficiencies in features and labels. In this chapter we propose a solution to the problem of simultaneous action classification using the MC framework (introduced in the previous chapter) addressing the issues of missing features and labels. With MC method of classification the problem would be still supervised with reduced training size. We propose *deep features* learned from a convolutional neural network to represent videos. Brief background on deep learning and convolutional neural networks follows in the next Section 5.2 .

5.2 Deep Learning

Deep learning is a branch of machine learning which has evolved from neural networks and gained immense attention recently. The primary concern of

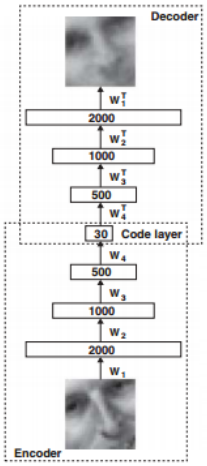

Figure 5.2: Deep autoencoder
The number of hidden units decrease till the code layer and increase to the output layer for reconstruction. The output from the code layer are used as features representing the given input. The encoder section converts the given input to a low-dimensional features and the decoder section uses these features to reconstruct the input [125]

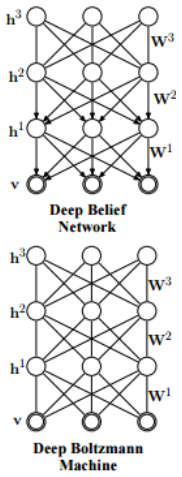

Figure 5.3: DBN and DBM
Top part of the figure is DBN showing the directed connections and the bottom one is DBM showing the undirected connections with 3 hidden layers h^1, h^2, h^3 and a visible layer 'v'. Figure adapted from [127]

deep learning methods is to move closer towards building true Artificial Intelligence (AI) systems. These methods tend to *automatically* learn powerful non-linear representations of the data in a hierarchical manner unlike conventional hand-crafted couterparts and have shown to beat the state-of-the-art in text [121], audio [122] [123] and image applications [124]. There are generative models which learn features from input in an unsupervised manner like Deep Autoencoders [125] , Deep Belief Nets DBNs [126] and Deep Boltzmann Machines DBMs [127]. Autoencoders are adaptive multilayer networks which map high-dimensional data into a low-dimensional features in a non-linear fashion. The low-dimensional features are learned by minimizing the error between the actual data and its reconstruction. Hence the input and output layers are the actual data and the features are intermediate outputs from one of the hidden layers. DBNs are generative, non-convolutional deep networks which have shown to outperform kernelized SVM on MNIST digit dataset [121]. DBMs are probabilistic graphical models of stacked layers of Restricted Boltzmann Machines (RBM)s like DBN with different connections between the layers. The connections between layers in DBNs are directed whereas they are undirected in DBMs [127].

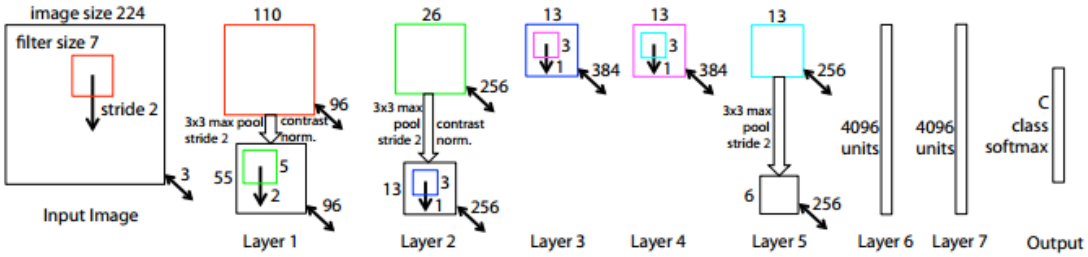

Figure 5.4: CNN for Image classification

CNN from [130] consists of 7 layers excluding input and output layers. Layers 1,2 are composed of convolution, pooling and contrast normalization operations. Layers 3,4,5 are composed of convolution and pooling. Layers 6 and 7 are fully connected layers followed by a softmaxloss operator where C is the number of classes.

5.2.1 Convolutional Neural Networks - CNNs

CNNs or *Convnets* [128] are discriminative models for deep learning. CNNs on large datasets like Imagenet [129] have shown to outperform DBNs. CNNs are more advanced extensions of fully connected neural networks which learn features from input at multiple levels in a supervised manner i.e. from labelled data. The learning process is achieved through online backpropagation [121] with gradient descent optimization. The supervised learning of CNNs results in discriminative features such that they cluster according to the classes. This makes CNNs well suited for classification and recognition tasks. A basic CNN is comprised of one or more blocks of convolution, rectification, pooling or sub-sampling operations (some applications include normalization operation also). In some papers each block is referred to as layer, in others each set of all or some operations are referred to as layers. We follow the former notation in this chapter. Figure 5.4 shows the convnet from [130] for image classification where the latter notation is used. We briefly give the details of the layers in a CNN adapted from [131].

Convolution layer

Convolutional layer consists of a filter bank, with trainable parameters or weights. Each filter (normally known as the local receptive field) adapts to a specific set of features from the input in the learning process and the output of the filter is known as *feature map*. For each input, the convolutional layer produces as many number of feature maps as the number of local receptive fields (filters). Let v_i be an input video, w_k be the k_{th} receptive field with dimensions (m, n, p). A

spatio-temporal point (x, y, t) of the k_{th} feature map v_k is given by

$$v_k^{xyt} = \sum_m \sum_n \sum_p w_k^{mnp} v_i^{(x+m)(y+n)(t+p)} \tag{5.1}$$

The parameters w are shared within each feature map and hence known as shared weights.

Rectification

These feature maps are then applied to a non-linear gating operator through rectification layer. The rectification is more biologically inspired to mimic human brain's neurons' activity, to fire on certain inputs only. Earlier sigmoid or tanh functions were used for rectification whose range is within $[0, 1]$ or $[-1, 1]$. The gradient vanishes if the input is increased or decreased. *Relu* (*Re*ctified *L*inear *U*nit) has substituted for the conventional sigmoid or tanh funtions.

$$f(\mathbf{x}) = \mathbf{max}(0, \mathbf{x}) \tag{5.2}$$

Relu applies a *'Max'* function to the input which is a simple thresholding when compared to the sigmoid and tanh. This also improves the convergence rate around 6 times [124].

Pooling

Pooling is a *downsampling* operator which producing translational invariance to the features. A small neighbourhood of a point is considered and some kind of statistics are applied within that neighbourhood. This patch/cube is replaced with the result obtained from the statistics. Typically *average* (*mean*) or *max* pooling is used in classification or recognition tasks. Max pooling replaces with the maximum value of all the points in the patch/cube. Let a pooling window be defined with dimension (x', y', t') then max pooling is given by

$$v_{xyt} = \max(v_{x'y't'} : x \leq x', y \leq y', t \leq t') \tag{5.3}$$

Average pooling replaces a small patch with the average of the pixels/points within the patch.

Backpropagation

These three operations are followed by dense (*fully connected*) layers which are connected to all activations in the previous layer. The output of the fully connected layers would be the class scores. Such network is trained using the standard *online backpropagation* algorithm introduced in [121]. CNN learns the features by minimizing an empirical loss function computed over all the training samples. The loss function of the CNN is given by Equation 5.4 adapted from [132]

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} l(\mathbf{y}_i, \boldsymbol{f}(\mathbf{v}_i; \mathbf{w})) \tag{5.4}$$

where $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ . \ . \ , \mathbf{w}_L)$ are the parameters of the entire CNN with L layers, $\mathbf{v}_i$ and $\mathbf{y}_i$ are the input and output of the CNN. Most commonly, the loss is minimized using *Gradient Descent* approach. First the gradient of the objective $\mathcal{L}$ is computed with current $\mathbf{w}$. Using the gradient, $\mathbf{w}$ is updated along the steepest descent direction. The update rule is given by Equation 5.5.

$$\mathbf{w}^t = \mathbf{w}^{t-1} + \eta_t \frac{\partial \boldsymbol{f}}{\partial \mathbf{w}}(\mathbf{w}^{t-1}) \tag{5.5}$$

where η_t is the learning rate. Computation of the gradient of the function $\boldsymbol{f}$ directly is hard since it is composite function and is computed using chain rule. Following the same notation as [132], the gradient of $\boldsymbol{f}$ with respect to $\mathbf{w}_l$ can be written as

$$\frac{\partial \boldsymbol{f}}{\partial \mathbf{w}_l} = \frac{\partial}{\partial \mathbf{w}_l} \boldsymbol{f}_L(\ . \ . \boldsymbol{f}_2(\boldsymbol{f}_1(\mathbf{v}; \mathbf{w}_1); \mathbf{w}_2). \ .); \mathbf{w}_L) \tag{5.6}$$

where l is the layer number. To compute the overall gradient using chain rule, the gradient of each $\boldsymbol{f}_l$ with respect to $\mathbf{w}_l$ and $\mathbf{v}_l$ must be computed first. The required $\dfrac{\partial \boldsymbol{f}}{\partial \mathbf{w}_l^T}$ is given by

$$\frac{\partial \boldsymbol{f}}{\partial \mathbf{w}_l^T} = \frac{\partial g_{l+1}}{\partial \mathrm{vec}\mathbf{v}_{l+1}^T} \frac{\partial \mathrm{vec}\boldsymbol{f}_l}{\partial \mathbf{w}_l^T} \tag{5.7}$$

where the notation $\dfrac{\partial \mathrm{vec}\boldsymbol{f}_l}{\partial \mathbf{w}_l^T}$ implies the derivative of a column vector of output to row vectors of parameter matrix $\mathbf{w}$ and g_{l+1} is the composite function from $l+1$ layer to last layer L of CNN. Now the main iteration would be computing the derivatives of l layer given the derivatives from $l+1$ layer. Backpropagation, thus involves in computing the derivatives from the last layer which means for a given layer l the input is $\dfrac{\partial g_{l+1}}{\partial \mathbf{v}_{l+1}^T}$ and the output would be $\dfrac{\partial g_l}{\partial \mathbf{v}_l^T}$ and $\dfrac{\partial g_l}{\partial \mathbf{w}_l^T}$.

5.3 Deep Action Classification

MC and deep learning techniques have been successfully applied to various vision problems dealing with images. Fusing both techniques, we propose robust classification of multiple actions against missing features and labels, with each action represented using learned deep features, within the matrix completion framework. Under the setting where the unknown entries are only the test labels, the problem can be formulated as Equation 4.5 and 4.6 as mentioned in the previous chapter. With some unobserved features and labels, the problem statement remains same but the constraints are with respect to only observed features and labels. We formalize the classification problem as matrix completion problem first and then justify our choice of deep features for the proposed framework. Following the same notations we can write

$$\begin{aligned} &\text{minimize} \quad \mu\|\mathbf{A}\|_* + l_X(\mathbf{A_X}) + \lambda\, l_Y(\mathbf{A_Y}) \\ &\text{subject to} \quad \mathbf{A_X} \geq \mathbf{0} \;\text{ and }\; \mathbf{A_1} = \mathbf{1}^{\mathrm{T}} \end{aligned} \tag{5.8}$$

where

$$\begin{aligned} l_X(\mathbf{A_X}) &= \sum_{ij \in \mathbf{\Omega_X}} (\mathrm{a}_{ij} - \mathrm{a}_{0ij})^2 \\ l_Y(\mathbf{A_Y}) &= \sum_{ij \in \mathbf{\Omega_Y}} \frac{1}{\gamma} \log(1 + \exp(-\gamma \mathrm{a}_{ij} \mathrm{a}_{0ij})) \end{aligned} \tag{5.9}$$

and $\mathbf{A_X} = [\mathbf{X}_{\mathrm{tr}} \;\; \mathbf{X}_{\mathrm{tst}}]$ is the feature sub-matrix, $\mathbf{A_Y} = [\mathbf{Y}_{\mathrm{tr}} \;\; \mathbf{Y}_{\mathrm{tst}}]$ is the label sub-matrix, $\mathbf{A_1} = \mathbf{1}^{\mathrm{T}}$ is used to handle bias and thus $\mathbf{A} = [\mathbf{A_Y}; \mathbf{A_X}; \mathbf{A_1}]$. $\mathbf{\Omega_X}$ is the set of observed feature indices and $\mathbf{\Omega_Y}$ is the set of observed label indices. This is a transduction classification problem in the semi-supervised learning setting for multi-label classification [133]. In that the correlation between labels aids the underlying low rank structure. For multi-class classification, the labels are distinct and hence the problem remains supervised with MC framework with reduced training size. In such cases, the low rank structure predominantly relies on features. For this reason, we train a CNN to learn features which are common within a specific class and distinct from other classes. The trained CNN will output features from the samples, which will be clustered according to the classes. We refer to these learned features as *deep features*. The prime aspect of convnets is their ability of *transfer learning*. There are several pre-trained networks available which can be used to extract pertinent features by fine-tuning the last layers. The available pre-trained models in the literature are suitable for image-based classification tasks since the convolution and pooling operations are spatial. To extract spatio-temporal features, these operations must extend in 3D in multiple

channels as in [21] and [22]. Typically, for classification tasks a classifier like linear SVM is again trained on the CNN features.

5.4 Experiments

Our complete experiment setup can be explained in two parts - (1) Training a CNN and extracting deep features (2) Classification with MC using deep features. Our experiments are performed on KTH dataset [110], introduced in the previous chapter. The performance of the proposed approach with deep features is compared to the state-of-the-art trajectory features.

5.4.1 CNN training and deep features

In this section, details of training the CNN to extract deep features is explained. The main steps include (i) preprocessing data before inputting to the CNN (ii) Designing the structure of CNN (iii) choosing the learning parameters.

Data preprocessing

Various preprocessing techniques have been adopted in the literature. For example, human centric bounding boxes are extracted and are contrast normalized in [22]. Foreground is extracted from the frames, gradient in x and y and optical flow in x and y are computed in [21]. All these five hand-crafted features are input to CNN. Also in [134], a two stream 2D CNN is proposed to handle the spatial and temporal information separately using optical flow frames as input to temporal stream.

In our work, the dataset is divided into training and test sets as in [110], out of 25 persons, videos of 16 persons are used for training and the remaining 9 for testing. For our experiments, the frames are resized to 60×80 from 120×160 and a sequence of 13 frames is considered. A video is represented as $60 \times 80 \times 13$ spatio-temporal cube with overall mean subtracted from each video. Apart from the resizing, no other preprocessing is made on the data. The performance of CNN can be enhanced if trained on large data, so typically the training data is expanded by including jittered or distorted copies of original data [135]. The training size is expanded 3 times by adding horizontally and vertically flipped frames.

CNN structure

The CNN is built using 3D convolution and pooling layers using the toolbox from [136]. These are compatible with Matconvnet toolbox [137]. We use the *Relu* layer from Matconvnet. We start with a CNN architecture similar to [21] and [22] but not exactly same. Denoting a set of convolution, rectification and pooling layers as a single *stage*, our initial architecture consisted of 3 stages. Stage 1 has a filter bank of 56 with the size of each convolution filter being $7 \times 7 \times 5$. Stage 2 and 3 result in 64 and 72 feature maps respectively with each filter of size $5 \times 5 \times 3$. The three stages are followed by two convolution layers which produce 84 feature maps each and the filter sizes are $3 \times 3 \times 3$ and $1 \times 4 \times 2$. The output of the last convolution layer is 84×1 and is fully (densely) connected to 6 neurons, each resulting in a class score. The pooling is max-pooling with a window size is $2 \times 2 \times 1$ in all the 3 stages. The convolution stride for all convolution layers is fixed as 1 spatially and temporally. The pooling stride is 2 spatially and 1 temporally.

The training parameters weight decay, momentum are set to 0.005 and 0.9 respectively. Initially the learning rate η is chosen as $1e-4$ for 20 epochs and later decreased to $1e-5$. The network is trained on the training data online backpropagation algorithm with weight sharing and momentum from [121]. The training is stopped after 60 epochs. The output of the last convolution layer before the class scores are considered as features of dimension 84×1. Though the objective has sufficiently converged, the overall classification accuracy with MC using these features was 83.78%. To improve the accuracy, the number of feature maps in each layer is increased first and decreased the convolution filter sizes. The smaller filter sizes have proved to be the best with image classification as well [138]. Four configurations of CNN (convnet) are considered, varying the feature maps and the filter sizes as shown in Figure 5.5. Each is trained using same set training parameters, mentioned above. The number of epochs required for training in each case was between 30 to 40.

Among the four, the *deeper* CNN with smallest filter sizes - Convnet 4 -performs the best. The elaborate structure of the Convnet 4 is shown in Figure 5.6. Deep features are extracted from the second last FC layer of this configuration which are of dimension 1024. For the first part in our experiments, the best recognition rate with our approach is **90.28%** where multiple test samples are jointly classified. This is comparable to the state of the art of 92.17% in [22] or 90.20% in [21] on the same dataset where CNN features are used. After training to see if the

Convnet 1	Convnet 2	Convnet 3	Convnet 4
C – 7x7x5 – 56	C – 7x7x5 – 96	C – 5x5x5 – 96	C – 3x3x3 – 96
Relu			
Max pooling – 2x2x1			
C – 5x5x3 – 64	C – 5x5x3 – 128	C – 5x5x3 – 128	C – 3x3x3 – 128
Relu			
Max pooling – 2x2x1			
C – 5x5x3 – 72	C – 5x5x3 – 256	C – 3x3x3 – 256	C – 3x3x3 – 256
Relu			
Max pooling – 2x2x1			
C – 3x3x3 – 84	C – 3x3x3 – 512	C – 3x3x3 – 512	C – 3x3x3 – 512
C – 1x4x3 – 84	C – 1x4x3 – 1024	C – 1x2x3 – 1024	C – 1x3x5 – 1024
FC – 1024	FC – 1024	FC – 1024	FC – 1024
Loss	Loss	FC – 1024	FC – 1024
-	-	Loss	Loss

Figure 5.5: Different configurations of CNN architectures explored.

Figure shows four configurations of CNNs (convnets) considered, varying the number of feature maps and filter sizes to improve the classification accuracy. (C - $i \times j \times k$ - N) represents 3D convolution layer where i , j are the spatial dimensions, k is the temporal dimension and N is the number of feature maps of that layer. Relu is the rectification layer. FC represents a Fully Connected network. Loss is Softmax loss function to compute the error between labels and class scores.

Configuration	Feature Dimension	Recognition rate(%)
Convnet 1	84	83.78
Convnet 2	1024	86.12
Convnet 3	1024	89.4
Convnet 4	1024	**90.28**

Table 5.1: Recognition rate with different configurations of CNNs.

The above Table reflects the improvement in performance of the CNN by increasing the number of feature maps of convolution layers and reducing the filter sizes.

CNN has learnt meaningful features, we have extracted some feature maps from the first convolution layer which actually produces 96 feature maps as shown in Figure 5.6. For four actions, boxing, handclapping, handwaving and running, a random frame is considered. For the same frame it is seen that each feature map emphasizes a particular feature. Out of 96, Figure 5.7 shows 4 feature maps.

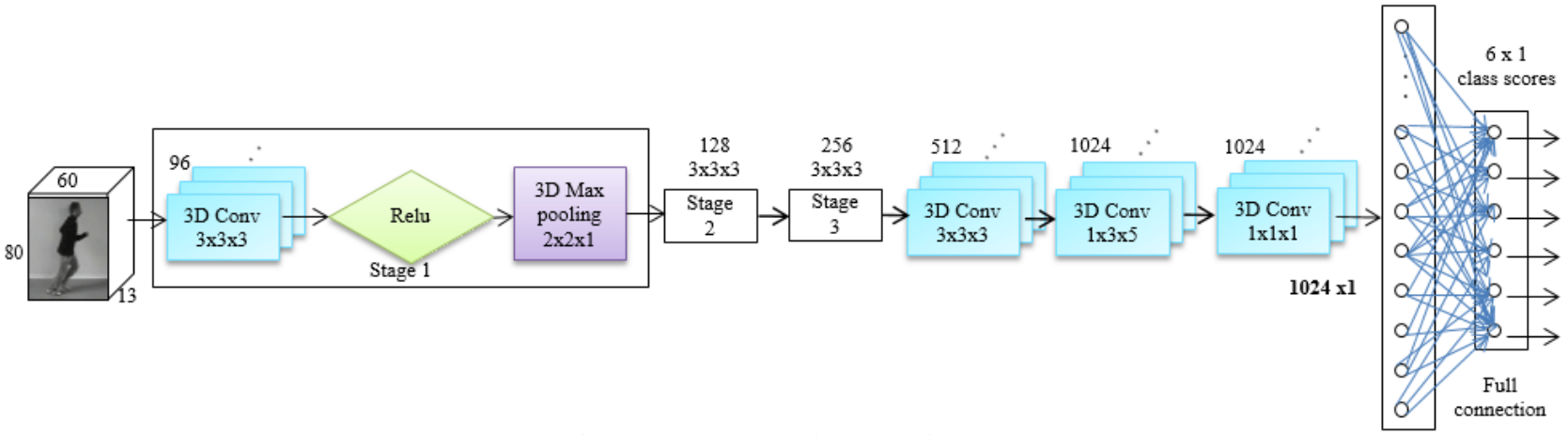

Figure 5.6: CNN architecture for deep feature extraction.

It has 3 stages with each stage consisting of convolution, rectification and pooling layers. The 3 stages are followed by two convolution layers. The output of the last convolution layer is fully connected to activation units to compute the class scores. CNN is trained by backpropagating the error from the class scores. After training, the output from the last convolution layer are considered as features.

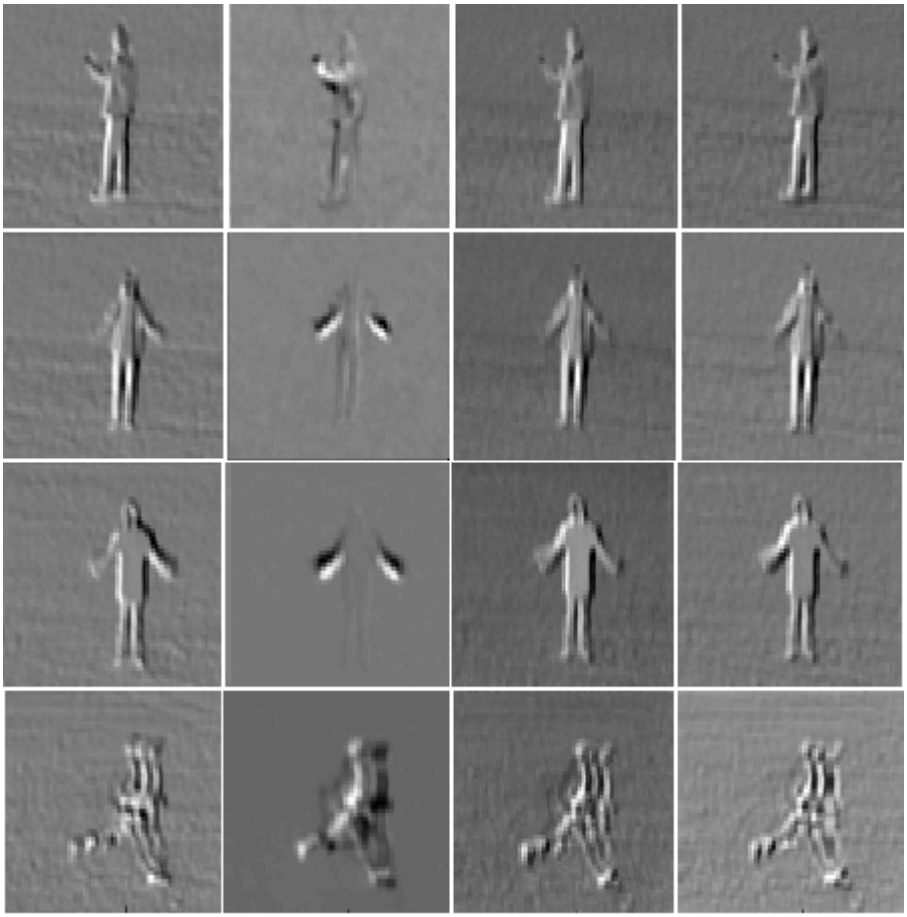

Figure 5.7: Feature maps from the first convolution layer

The above figure shows the feature maps 1, 10, 35 and 75 from the first convolution layer in the CNN 5.6. Each row corresponds output to same frame of an action. Feature maps of four actions boxing, handclapping, handwaving and running are shown. It is seen that each map is reflecting distinct features such as shape or motion or both.

5.4.2 Dense trajectory based features

Dense trajectory feature descriptors proposed by Wang et.al. in [47] are extracted towards the second part of experiments. The procedure of extracting these features is explained in Chapter 4, Section 4.4.3. In this chapter, for each trajectory, we compute only MBH descriptors as these alone have shown to outperform remaining descriptors on this dataset [47]. Each descriptor is 192 with 96 in x and 96 in y directions. This is reduced by two using PCA. The reason for using PCA is explained in Chapter 4, Section 4.4.3 . A GMM with $K = 256$ is learnt on the PCA reduced descriptors from training data. Each of these descriptors are fisher-vector encoded using the GMM. The dimension of the fisher encoded MBH descriptor is 49152. We have further reduced the dimension with RP to 3072.

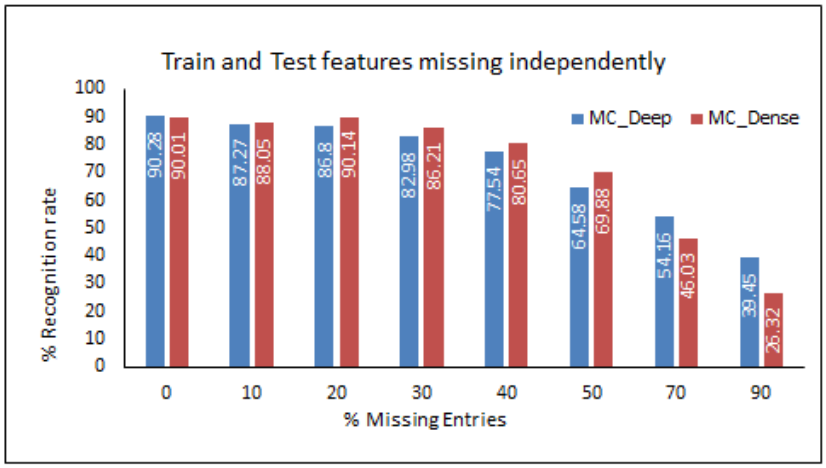

Figure 5.8: MC results on missing train and test features independently

Two independent random masks are created and applied on train and test feature matrices, $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$. The train labels are complete. The accuracy with both features decreases with increase in missing data. It is also observed that beyond 50% of the data missing, the decline is more pronounced with dense features than deep features. MC_Deep denotes the performance of deep features using MC framework and MC_Dense denotes the performance of dense features using the same MC framework.

5.4.3 Classification with MC

Feature sub-matrix $\mathbf{A}_{\mathbf{X}}$ is constructed using the features of train data and test data. $\mathbf{X}_{train}$ and $\mathbf{X}_{test}$ are of size 1024×1524 and 1024×144 respectively with deep features and 3072×1524 and 3072×144 with dense features. $\mathbf{Y}_{train}$ and $\mathbf{Y}_{test}$ are of size 6×1524 and 6×144. All the unknown or missing entries are set to 0 initially. The parameters are set to $\lambda = 0.01$, $\gamma = 30$, reduction parameter to compute set of μ values, η_μ is set to 0.25 and the final least value of μ is set to $1e-8$ as in the previous chapter. MC method is evaluated under three scenarios - (1) Randomly missing training and test features with independent masks (2) Randomly missing training and test features jointly (3) Randomly missing features and labels jointly. The observations recorded in all three cases are shown in Figure 5.8 , Figure 5.9 and Figure 5.10 respectively.

Class	boxing	handclap	handwave	jogging	running	walking	mAP
Case 1	0.9271	0.9626	0.9513	0.8142	0.9212	0.8502	0.9045
Case 2	0.9905	0.9090	0.9722	0.8905	0.8590	0.8323	0.9089
Case 3	0.8967	0.9545	0.9305	0.8048	0.8714	0.8284	0.8811
Case 4	0.7809	0.8219	0.8538	0.6260	0.6144	0.7202	0.7362

Table 5.2: mAP with deep and dense features with no missing data and 70% missing data.

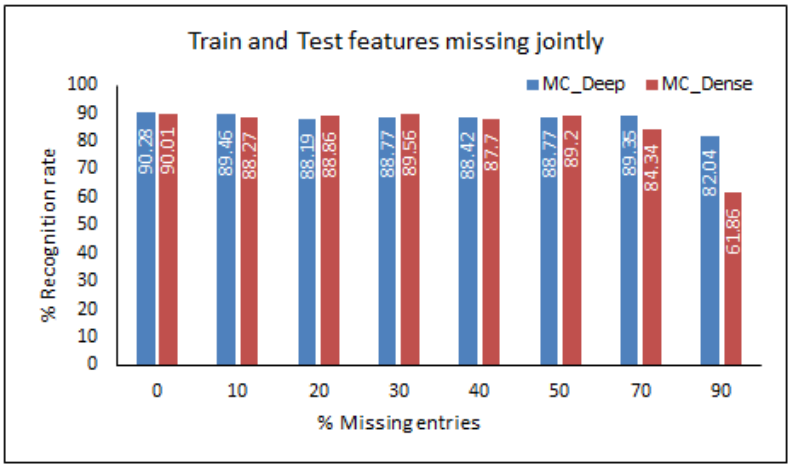

Figure 5.9: MC results on missing train and test features jointly

A single random mask is created and applied on entire feature matrix $\mathbf{A_X}$ which consists of Train and Test features. No mask is applied on $\mathbf{A_Y}$ which means Train labels are complete and Test labels are missing. The accuracy with deep features is more stable upto 70% of missing data than the previous case. MC_Deep denotes the performance of deep features using MC framework and MC_Dense denotes the performance of dense features using the same MC framework.

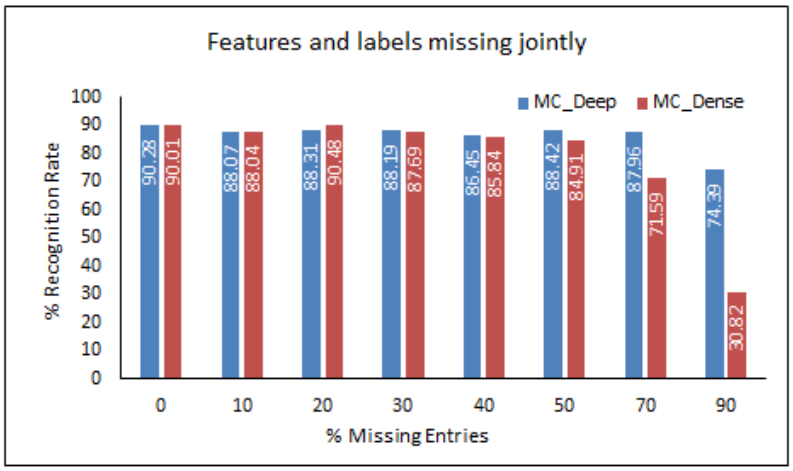

Figure 5.10: MC results on missing train and test features jointly

For the third scenario, train and test features $\mathbf{A_X}$ and train labels $\mathbf{Y}_{train}$ are jointly missed using a single random mask. It is noted that when more than 50% of the data missing, the accuracy with dense features decreases at a faster rate than that of deep features. MC_Deep denotes the performance of deep features using MC framework and MC_Dense denotes the performance of dense features using the same MC framework.

5.4.4 Discussion

In all the three cases, it is observed that the MC with deep features performs better than with dense features when the percentage of missing entries is significant. The joint low rank structure assumption is stronger when the entries

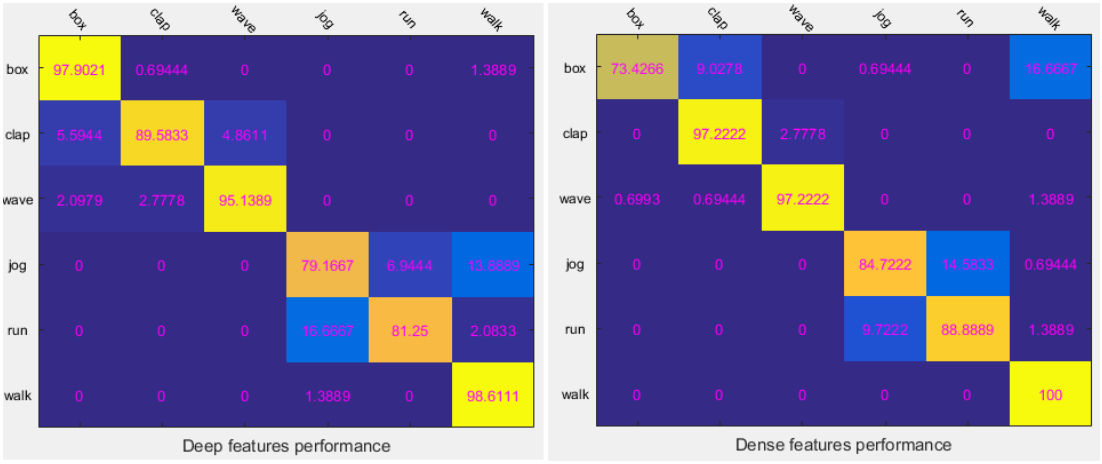

Figure 5.11: Confusion matrices for deep and dense features with no missing data

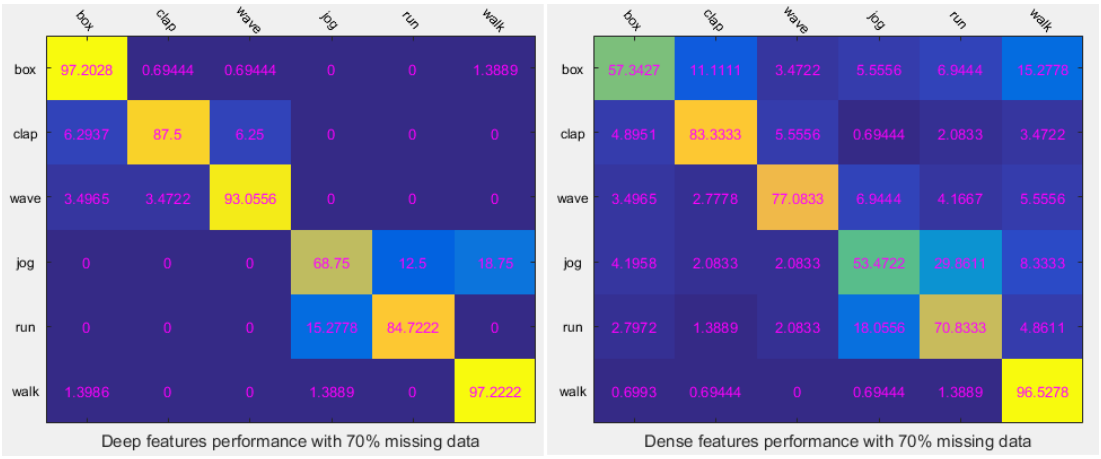

Figure 5.12: Confusion matrices for deep and dense features with 70% missing data

are removed using a single mask as in second and third cases which are more generic scenarios. In the second case, performance with deep features is 5% more than dense features with 70% missing data and in the third case the difference is more than 15% for the same percentage of missing data. The performance is also validated using confusion matrices shown in Figure 5.11, Figure 5.12 and computing mAP shown in Table 5.2. The confusion matrices with no data missing and 70% randomly missing features and labels jointly are shown in Figure 5.11 and Figure 5.12. The mean Average Precision mAP for 4 cases is shown in Table 5.2. In that table, 'Case 1'denotes the case with deep features with no missing data, 'Case 2'denotes dense features with no missing data. 'Case 3'denotes deep features with 70% of features and labels missing jointly and 'Case 4'denotes dense features with 70% of features and labels missing jointly. An important consideration in assessing these features is the quality of raw data on which the these are

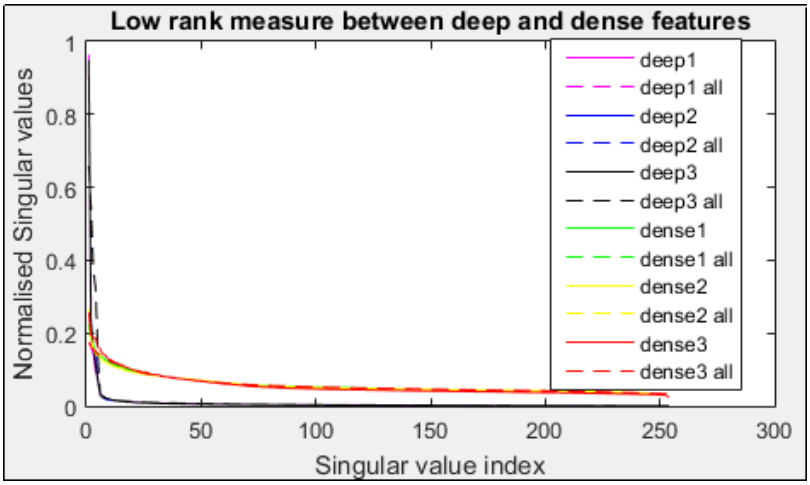

Figure 5.13: Visualization of the distribution of singular values with deep and dense features.

Figure showing the distribution of normalised singular values of feature matrices $\mathbf{X}_{cl}$ and $\mathbf{X}_{all}$ for three classes - boxing(1), handclapping(2) and handwaving(3). The notation in the above is 'deep1'and 'deep1 all'correspond to $\mathbf{X}_{cl}$ and $\mathbf{X}_{all}$ with deep features of class 1 boxing and 'dense1'and 'dense1 all'correspond to the same with dense features of class 1, boxing. Similarly 'deep2', 'deep2 all','dense2', 'dense2 all'correspond to those of class 2, handclapping, and 'deep3', 'deep3 all','dense3', 'dense3 all'correspond to those of class 3, handwaving. The decay of normalised singular values of deep features is more sharper than that of dense features.

extracted. While dense features are extracted from a clean video and at different spatial scales, deep features in this work are extracted from substantially downsized videos with comparable and improved performance at various levels of data abstraction.

Low rank measure

To qualitatively assess these features in the context of MC, we compute Nuclear Norm Ratio NNR as an empirical rank measure [106]. NNR is defined as the ratio of nuclear norm of a feature matrix with features from a single class to the feature matrix formed with features from all the classes. Let $\mathbf{X}_{cl}$ be the feature matrix pertaining to a class and $\mathbf{X}_{all}$ be the feature matrix with features from all classes in the dataset with same number of samples as in $\mathbf{X}_{cl}$. Nuclear Norm Ratio NNR can be written as Equation 5.10.

$$\text{NNR} = \frac{\|\mathbf{X}_{cl}\|_*}{\|\mathbf{X}_{all}\|_*} \tag{5.10}$$

For the complete matrix to be of low rank, this ratio would be less than 1. For deep and dense features, for each class NNR is computed and is tabulated in Table

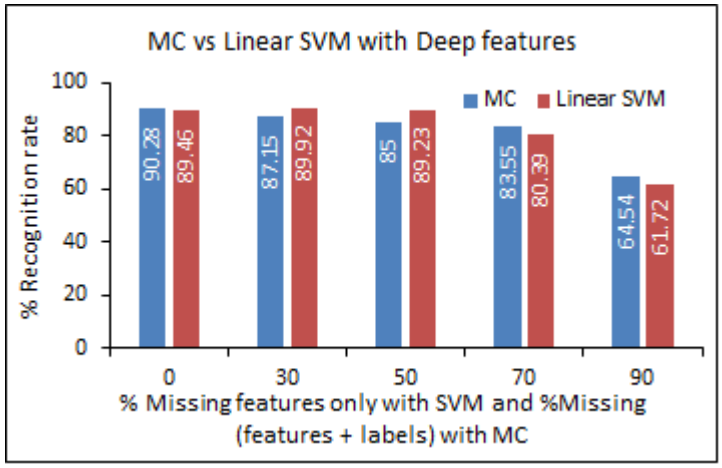

Figure 5.14: Performance comparison between MC and SVM with deep features

5.3. These values must not be confused with the recognition rates. Recognition rate is computed as the ratio of overall true positives in each class to total number of samples. For the $\mathbf{X}_{cl}$ and $\mathbf{X}_{all}$ with dense features, we considered the original

Class	Deep features	Dense features
boxing	0.7812	0.9899
handclapping	0.7832	0.9901
handwaving	0.777	0.95
jogging	0.8938	0.999
running	0.8145	1.0150
walking	0.6746	0.9495

Table 5.3: NNR for all the classes in KTH dataset

The table above shows the NNR computed with deep and dense features for all the classes in KTH dataset. It is seen that this is lower in all classes with deep features than the corresponding dense features. Also for class running NNR greater than 1 with dense features.

features without RP. For a visual representation of the singular value distribution in both cases, we plot the same in Figure 5.13. The decay of the singular values is faster and is very close to zero with deep features than that of dense features. This validates that the low rank assumption of the complete matrix with deep features is stronger and thus the classification is reasonably well even when more than 70% of the features are missing.

Comparison with SVM

Our approach is compared with linear SVM classifier from [92]. Although this is not a fair comparison as the SVM is trained everytime for each percentage of

missing entries. The unknown features were replaced with zeros. The results were found to be encouraging proving the robustness of MC at significant percentages of missing entries. In this case alone the entries were missed cumulatively. The comparison is shown in Figure 5.14

Limitations

The efficiency of the deep features relies of CNN training which in turn depends on number of the training data samples. Larger training datasets result in more generic features. The number of parameters generated with the CNN structure employed in our work, are 42168 which is very less compared to 2 million parameters approximately in [21] but slightly greater than 17169 of [22] . All these parameters have to be stored once the CNN is trained. The main setback of the entire approach is, all the training features must be stored since MC methods are applied on the entire feature set. This puts a limitation on the feature size. The computation time of running the MC algorithm is mainly dependent on the SVD step which is limited by the feature dimension.

5.5 Conclusion

In this chapter, we presented an approach for joint classification of multiple actions represented with deep features using matrix completion robust to feature and label deficiencies. Proposed deep features are extracted efficiently by training a CNN without any complicated preprocessing as in other works. From the experiments it is observed that the performance of MC with deep features is stable than dense features in various settings of missing features and labels. The performance deteriorates when 70% or more data is missing. The heuristic rank measure also confirms that the matrix with deep features preserves the joint low-rank structure of the matrix with features and labels and the experiments validate this even under extreme conditions.

Chapter 6

Conclusion

This is the final chapter of our thesis. We start with a review from where and how we have started our approach to the solve the problem of action classification. Key contributions towards the proposed goal are summarized. Some possible future extensions are also discussed.

6.1 Summary

This thesis has started out with a goal to design a robust action recognition system exploiting the inherent redundancy in video data to identify multiple actions simultaneously. Towards realising this, we focussed on exploiting structures within data to facilitate classification. Specifically, the applicability of two main paradigms of CS, sparse and low rank approximations of data in feature space, to the problem of action classification has been explored. Towards the first part of our goal, sparsity is exploited in the compressed feature space for classification in Chapter 3. Compressed features obtained using random projection have shown to excel, achieving state of the art performance even with feature dimension as low as 72 compared to the actual dimension of 9600 with GEIs. The performance is better than state of the art [17–19] in action recognition with sparse representation.

The notion of sparsity is extended to low rank structures in Chapter 4. Applying the principles of linear classification, the matrix formed with concatenation of features and labels is of low rank [105]. Based on this assumption, features and labels from train and test data are arranged as an incomplete matrix with unknown test labels. Minimizing the rank of this matrix, completes the test labels. The method is evaluated on three different datasets. The method achieved in **95%** and **93%** recognition rates on Weizmann and KTH datasets and **69%** on UCF101 dataset. The achieved results are better than or comparable to traditional approaches.

Matrix completion framework is extended to handle missing feature and label scenarios in Chapter 5. The underlying low rank structure of the feature-label matrix relies predominantly on features for multi-class classification. To this end, deep features are proposed. Deep features are extracted from a convolutional neural network which is trained on the data in a supervised fashion. These features have shown consistent performance even in the situations where more than 70% of the features are missing randomly. The best recognition rate with no missing data, except for the test labels, is **90.28 %** on KTH dataset and is very close to the state of the art, which is 92.17% from [22] and 90.20% from [21], using CNN features with minimal preprocessing of the data.

Dimensionality reduction using random projection proved to be an efficient method achieving high recognition rates even at very low projected dimensions. In all the

cases, the training stage required by a typical classifier like SVM is bypassed. The results achieved are much better than or comparable to a trained SVM.

6.1.1 Contributions

We consolidate our key contributions below towards our proposed goal.

1. An approach to classify actions and gestures using *compressed* features exploiting sparse structures of the features over a dictionary. [23]
2. A novel approach to classify multiple actions jointly using matrix completion is introduced. [24]
3. Deep features from a convolutional neural network are introduced for action classification with matrix completion method extended to deal with feature and label deficiencies. The performance with deep features have proven to beat the state of the art [21, 22], using the low rank matrix completion setting. The recognition rate with proposed matrix completion framework is more consistent than that with SVM classifier.

6.2 Future aspects

We affirm that this thesis has made reasonable contributions towards realising the goal with which it has started. Taking from here, this work can be easily extended in the following directions. In this work we applied our methods on videos with one action performed by a single human. Handling the situations with multiple humans performing multiple actions would be extremely challenging requiring fusion of segmentation methods into models. Below we give some possible extensions this work in future.

Sparse representation based action classification with corrupted features

We have shown that a dictionary constructed from simple concatenation of features is efficient for computing sparse solution of test data for classification. For this compressed features were used as columns in the dictionary. When the features are grossly corrupted, the linear model Equation 3.1 or Equation 3.9 does not hold. In that case the dictionary must be concatenation of *feature basis* and the basis in which the error due to corruption is sparse [11]. What is challenging here is the prior knowledge of error basis. Further the same could be extended to corrupted data itself rather than the corruption in feature space.

Matrix completion

In this thesis we have applied matrix completion with deep and dense features from videos with a single action under the feature deficit situations. This could be extended to study if the low rank approximation can still be used if some of the data itself has lost some information. This could happen in crowd surveillance scenarios.

Deep networks for action recognition

Features extracted from a trained convolutional neural network alone have shown to outperform dense features. Most recent works in action recognition combine these features explicitly and have achieved state of the art performances. The CNN used to train in these works is still the standard Alexnet [124] or Googlenet [128] which are trained on multi-channel images but not videos. Future work can address this by releasing similar networks trained on videos which learns spatio-temporal features. This requires lot of training data and computational resources to reach to the standards of the image based nets.

Bibliography

[1] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing Action at a Distance," in *Proc. of the 9th IEEE Int. Conf. on Computer Vision*, pp. 726–733, IEEE Computer Society, 2003.

[2] A. Kläser, M. Marszałek, and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients," in *British Machine Vision Conf.*, pp. 995–1004, sep 2008.

[3] M. Elad and M. Aharon, "Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries," *IEEE Transactions on Image Processing*, pp. 3736–3745, Dec 2006.

[4] J. Medina, "Brain rules." `http://www.brainrules.net/pdf`, 2014. Accessed : 2016-07-01.

[5] E. Candès and M. Wakin, "An Introduction to Compressive Sampling," *Signal Processing Magazine, IEEE*, pp. 21–30, March 2008.

[6] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of Compressed Sensing for Rapid MR Imaging," *Magnetic Resonance in Medicine*, pp. 1182–1195, 2007.

[7] M. Firooz and S. Roy, "Network Tomography via Compressed Sensing," in *Global Telecommunications Conference, IEEE*, pp. 1–5, Dec 2010.

[8] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, "Single-Pixel Imaging via Compressive Sampling," *Signal Processing Magazine, IEEE*, pp. 83–91, March 2008.

[9] J. Bennett, S. Lanning, and N. Netflix, "The Netflix Prize," in *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[10] S. Agarwal and D. Roth, "Learning a sparse representation for object detection," *European Conf. on Computer Vision*, pp. 97–101, 2006.

[11] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, no. 2, pp. 210–227, 2009.

[12] F. Rodriguez and G. Sapiro, "Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries," tech. rep., Defense Technical Information Center Document, 2008.

[13] A. Yilmaz and M. Shah, "Recognizing human actions in videos acquired by uncalibrated moving cameras," in *IEEE Int. Conf. on Comp. Vision*, pp. 150–157, Oct 2005.

[14] J. Aggarwal and M. Ryoo, "Human Activity Analysis: A review," *ACM Computing Surveys*, pp. 16:1–16:43, April 2011.

[15] D. Oneata, J. Verbeek, and C. Schmid, "Action and Event Recognition with Fisher Vectors on a Compact Feature Set," in *Proc. of Int. Conf. Comp. Vision*, pp. 1817–1824, Dec. 2013.

[16] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Comp. Vision and Image Understanding*, 2016.

[17] C. Liu, Y. Yang, and Y. Chen, "Human action recognition using sparse representation," in *IEEE Int. Conf. on Intelligent Computing and Intelligent Systems*, vol. 4, pp. 184–188, 2009.

[18] K. Guo, P. Ishwar, and J. Konrad, "Action Recognition in Video by Sparse Representation on Covariance Manifolds of Silhouette Tunnels," in *Proc. of Int. Conf. of Pattern Recognition*, pp. 294–305, 2010.

[19] T. Guha and R. Ward, "Learning Sparse Representations for Human Action Recognition," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, pp. 1576–1588, 2012.

[20] R. S. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino, "Matrix Completion for Multi-label Image Classification," in *Advances of Neural Information Processing Systems*, pp. 190–198, 2011.

[21] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," in *IEEE Int. Conf. on Machine Learning*, pp. 495–502, Omnipress, 2010.

[22] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Proc. of the Int. Conf. on Human Behavior Understanding*, pp. 29–39, 2011.

[23] S. Bomma, P. Favaro, and N. M. Robertson, "Sparse representation based action and gesture recognition," in *IEEE Int. Conf. on Image Processing*, pp. 141–145, 2013.

[24] S. Bomma and N. Robertson, "Joint Classification of Actions with Matrix Completion," in *IEEE Int. Conf. on Image Processing*, pp. 2766–2770, Sept 2015.

[25] R. Poppe, "A Survey on Vision-based Human Action Recognition," *Image Vision Comput.*, vol. 28, pp. 976–990, June 2010.

[26] S. Escalera, J. Fabian, P. Pardo, X. Baro, J. Gonzalez, H. J. Escalante, D. Misevic, U. Steiner, and I. Guyon, "ChaLearn Looking at People 2015: Apparent Age and Cultural Event Recognition Datasets and Results," in *The IEEE Int. Conf. on Comp. Vision Workshops*, December 2015.

[27] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing Human Actions: A Local SVM Approach," in *Proc. of the Pattern Recognition, 17th Int. Conf. Volume 3 - Volume 03*, ICPR '04, pp. 32–36, IEEE Computer Society, 2004.

[28] S. Johnson and M. Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation," in *Proc. of the British Machine Vision Conf.*, pp. 12.1–12.11, BMVA Press, 2010. doi:10.5244/C.24.12.

[29] A. F. Bobick and J. W. Davis, "The Recognition of Human Movement Using Temporal Templates," *IEEE Trans. Pattern Analysis and Machine Intelligence*, pp. 257–267, Mar. 2001.

[30] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as Space-Time Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, pp. 2247–2253, Dec 2007.

[31] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action MACH: A spatio-temporal Maximum Average Correlation Height filter for action recognition," in *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2008.

[32] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[33] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proc. of the 7th Int. Joint Conf. on Artificial Intelligence*, pp. 674–679, 1981.

[34] E. Shechtman and M. Irani, "Space-Time Behavior Based Correlation," in *IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 405–412, June 2005.

[35] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. of the 2005 IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 886–893, 2005.

[36] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning Realistic Human Actions from Movies," in *Int. Conf. on Comp. Vision and Pattern Recognition*, June 2008.

[37] G. Levi, "A Short introduction to descriptors." `https://gilscvblog.wordpress.com/2013/08/18/a-short-introduction-to-descriptors/`, 2013. Accessed: 2016-07-01.

[38] N. Dalal, B. Triggs, and C. Schmid, "Human Detection Using Oriented Histograms of Flow and Appearance," in *Proc. of the 9th European Conf. on Computer Vision*, ECCV'06, pp. 428–441, Springer-Verlag, 2006.

[39] I. Laptev, "On Space-Time Interest Points," *Int. Journal on Comp. Vision*, vol. 64, pp. 107–123, 2005.

[40] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[41] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior Recognition via Sparse Spatio-temporal Features," in *Proc. of the 14th Int Conf. on Comp. Communications and Networks*, pp. 65–72, 2005.

[42] G. Willems, T. Tuytelaars, and L. J. V. Gool, "An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector," in *Proc. of European Conf. on Comp. Vision*, pp. 650–663, 2008.

[43] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *British Machine Vision Conf.*, pp. 124.1–124.11, 2009.

[44] T. Tuytelaars, "Dense interest points," in *2010 IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 2281–2288, June 2010.

[45] G. Johansson, "Visual Motion Perception," *j-SCI-AMER*, vol. 232, pp. 76–88, June 1975.

[46] Y. Sheikh, M. Sheikh, and M. Shah, "Exploring the Space of a Human Action," in *Int. Conf. on Comp. Vision*, pp. 144–149, IEEE Computer Society, 2005.

[47] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action Recognition by Dense Trajectories," in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 3169–3176, IEEE Computer Society, 2011.

[48] X. Wang, L. Wang, and Y. Qiao, "A Comparative Study of Encoding, Pooling and Normalization Methods for Action Recognition," in *Proc. of Asian Conf. on Comp. Vision.*, ACCV'12, pp. 572–585, 2013.

[49] J. Macqueen, "Some methods for classification and analysis of multivariate observations," in *In 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, 1967.

[50] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *Proc. of European Conf. Comp. Vision*, ECCV'10, pp. 392–405, 2010.

[51] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *Proc. of Int. Conf. on Comp. Vision*, pp. 2929–2936, June 2009.

[52] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," in *Proc. of Int. Conf. Comp. Vision*, pp. 3551 – 3558, 2013.

[53] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher Kernel for Large-scale Image Classification," in *Proc. of European Conf. on Comp. Vision*, pp. 143–156, 2010.

[54] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[55] R. Poppe and M. Poel, "Discriminative human action recognition using pairwise CSP classifiers," in *Proc. of the IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pp. 1–6, IEEE Computer Society, September 2008.

[56] D. Weinland and E. Boyer, "Action Recognition using Exemplar-based Embedding," in *Proc. of Int. Conf. on Comp. Vision and Pattern Recognition*, pp. 1–7, 2008.

[57] V. N. Vapnik, *The nature of statistical learning theory.* Springer-Verlag New York, Inc., 1995.

[58] A. Zisserman, "Lecture notes on svm dual, kernels and regression." `http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf`. Accessed: 2016-10-11.

[59] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A Biologically Inspired System for Action Recognition," in *IEEE 11th Int. Conf. on. Comp Vision*, pp. 1–8, Oct 2007.

[60] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images," *Society of Industrial and Applied Math. Review*, pp. 34–81.

[61] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Conf. on Signals, Systems and Computers*, pp. 40–44 vol.1, Nov 1993.

[62] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, pp. 3397–3415, Dec 1993.

[63] R. A. DeVore and V. N. Temlyakov, "Some remarks on greedy algorithms," *Adv. in Computational Math.*, pp. 173–187, 1996.

[64] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic Decomposition by Basis Pursuit," *Soc. for Ind. and Applied Math. Review*, pp. 33–61, Aug 1998.

[65] D. L. Donoho, "For Most Large Underdetermined Systems of Linear Equations the Minimal 1-norm Solution is also the Sparsest Solution," *Communications on Pure and Applied Math.*, vol. 59, pp. 797–829, 2004.

[66] M. Zibulevsky and M. Elad, "L1-L2 Optimization in signal and image processing," *IEEE Signal Processing Magazine*, pp. 76–88, May 2010.

[67] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for Sparse Representation Modeling," *Proc. of the IEEE, Vol. 98, no. 6,*, pp. 1045–1057, June 2010.

[68] D. L. Donoho and M. Elad, "Optimally sparse representation in general (non-orthogonal) dictionaries via minimization," in *Proc. Natl. Acad. Sci. USA*, pp. 2197–2202, 2002.

[69] B. A. Olshausen and D. J. Field, "Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images," *Nature*, pp. 607–609, 1996.

[70] K. Engan, S. O. Aase, and J. Hakon Husoy, "Method of Optimal Directions for Frame Design," in *Proc. of IEEE Acoustics, Speech, and Signal Processing, 1999*, pp. 2443–2446, IEEE Computer Society, 1999.

[71] M. J. Fadili, J. L. Starck, and F. Murtagh, "Inpainting and Zooming Using Sparse Representations," *Computer Journal*, pp. 64–79, 2009.

[72] J. Starck, M. Elad, and D. Donoho, "Image decomposition via the combination of sparse representations and a variational approach," *IEEE Transactions on Image Processing*, pp. 1570–1582, Oct 2005.

[73] M. Elad, J. Starck, P. Querre, and D. L. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)," *Applied and Computational Harmonic Analysis*, pp. 340 – 358, 2005.

[74] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed Sensing MRI," *IEEE Signal Processing Magazine*, vol. 25, pp. 72–82, March 2008.

[75] K. Huang and S. Aviyente, "Sparse representation for signal classification," *Advances in Neural Information Processing Systems*, pp. 609–616, 2007.

[76] D. Cai, X. He, J. Han, and T. S. Huang, "Graph Regularized Nonnegative Matrix Factorization for Data Representation," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1548–1560, Aug 2011.

[77] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Supervised Dictionary Learning," in *Advances in Neural Information Processing Systems*, pp. 1033–1040, 2009.

[78] C. K. Chiang, C. H. Duan, S. H. Lai, and S. F. Chang, "Learning component-level sparse representation using histogram information for image classification," in *Proc. of Int. Conf. on Comp. Vision*, pp. 1519–1526, Nov 2011.

[79] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. of Comp. Vision and Pattern Recognition*, pp. 1794 –1801, June 2009.

[80] A. Y. Yang, R. Jafari, S. S. Sastry, and R. Bajcsy, "Distributed Recognition of Human Actions Using Wearable Motion Sensor Networks," *Journal of Ambient Intelligence and Smart Environments*, pp. 103–115, 2009.

[81] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *Proc. of Comp. Vision and Pattern Recognition*, pp. 1–8, June 2008.

[82] K. Mikolajczyk and C. Schmid, "An Affine Invariant Interest Point Detector," in *Proc. of the 7th European Conf. on Comp. Vision*, pp. 128–142, 2002.

[83] R. Collins, "Mean-shift blob tracking through scale space," in *IEEE Conf. on Comp. Vision and Pattern Recognition*, vol. 2, pp. 234–240, 2003.

[84] J. Han and B. Bhanu, "Individual recognition using gait energy image," *IEEE Tran. on Pattern Analysis and Machine Intelligence,*, pp. 316–322, 2006.

[85] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proc. of the 7th Int. Conf. on Knowledge Discovery and Data Mining*, pp. 245–250, ACM, 2001.

[86] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "The Johnson-Lindenstrauss lemma meets compressed sensing," *Preprint*, 2006.

[87] E. J. Candès, Y. C. Eldar, and D. Needell, "Compressed Sensing with Coherent and Redundant Dictionaries," *Applied and Computational Harmonic Analysis*, pp. 59 – 73, 2010.

[88] K. Koh, S. Kim, and S. Boyd, "A matlab solver for large-scale l1-regularized least squares problems." `https://stanford.edu/~boyd/l1_ls/`, 2007. Accessed on 2016-07-01.

[89] A. Rakotomamonjy, "Algorithms for multiple basis pursuit denoising," in *Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[90] P. Barattini, C. Morand, and N. Robertson, "A proposed gesture set for the control of industrial collaborative robots," in *IEEE Robotics and Automation Society*, pp. 132 –137, Sept 2012.

[91] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 2432–2439, IEEE, 2010.

[92] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, pp. 27:1–27:27, 2011. Accessed: 2016-07-01.

[93] Z. Lin, Z. Jiang, and L. S. Davis, "Recognizing actions by shape-motion prototype trees," in *Proc. of Int. Conf. on Comp. Vision*, pp. 444–451, Sept. 2009.

[94] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, pp. 13:1–13:19, Dec. 2015.

[95] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, pp. 76–80, Jan. 2003.

[96] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, 1936.

[97] E. J. Candès and B. Recht, "Exact Matrix Completion via Convex Optimization," *Foundations of Computational Math.*, pp. 717–772, 2009.

[98] M. Fazel, *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.

[99] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization," *Society of Industrial and Applied Math. Review*, pp. 471–501, 2010.

[100] E. J. Candès and Y. Plan, "Matrix Completion with Noise," *Proceedings of the IEEE*, pp. 925–936, 2009.

[101] Z. Liu and L. Vandenberghe, "Interior-Point Method for Nuclear Norm Approximation with Application to System Identification," *Society of Industrial and Applied Math. Journal on Matrix Analysis and Applications*, pp. 1235–1256, 2009.

[102] P. Biswas, T. C. Lian, T. C. Wang, and Y. Ye, "Semidefinite Programming Based Algorithms for Sensor Network Localization," *ACM Transactions on Sensor Network*, pp. 188–220, May 2006.

[103] R. Keshavan, A. Montanari, and S. Oh, "Matrix Completion from a Few Entries," *IEEE Trans. on Information Theory*, pp. 2980–2998, June 2010.

[104] J. F. Cai, E. J. Candès, and Z. Shen, "A Singular Value Thresholding Algorithm for Matrix Completion," *Society of Industrial and Applied Math. Journal on Optimization*, pp. 1956–1982, Mar. 2010.

[105] A. B. Goldberg, X. Zhu, B. Recht, J. M. Xu, and R. D. Nowak, "Transduction with Matrix Completion: Three Birds with One Stone," Advances of Neural Information Processing Systems, pp. 757–765, 2010.

[106] R. S. Cabral, F. De la Torre, J. P. Costeira, and A. Bernardino, "Matrix Completion for Weakly-supervised Multi-label Image Classification," *IEEE Tran. Pattern Analysis and Machine Intelligence*, pp. 121–135, 2015.

[107] M. Liu, Y. Luo, D. Tao, C. Xu, and Y. Wen, "Low-Rank Multi-View Learning in Matrix Completion for Multi-Label Image Classification," in *Proc. of Association for the Advancement of Artificial Intelligence*, pp. 2778–2784, 2015.

[108] M. Fan, D. Zhao, Q. Zhou, Z. Liu, T. F. Zheng, and E. Y. Chang in *Proc. of Annual Meeting of the Association for Computational Linguistics*, pp. 839–849, June.

[109] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," *Mathematical Programming*, pp. 321–353, 2011.

[110] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. of Int. Conf. on Pattern Recognition*, pp. 32–36, IEEE, 2004.

[111] K. Soomro, A. R. Zamir, and M. Shah, "UCF101 : A dataset of 101 human actions classes from videos in the wild." http://crcv.ucf.edu/data/UCF101.php, 2012. Accessed: 2016-07-01.

[112] A. Ng, "Mixtures of Gaussians and the EM Algorithm." http://cs229.stanford.edu/notes/cs229-notes7b.pdf. Accessed: 2016-07-01.

[113] C. M. Bishop, "Mixture models and the EM Algorithm." http://mlg.eng.cam.ac.uk/tutorials/06/cb.pdf. Accessed: 2016-07-01.

[114] A. Vedaldi and B. Fulkerson, "VLFeat - An open and portable library of computer vision algorithms." http://www.vlfeat.org/, 2010. Accessed: 2016-07-01.

[115] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," in *Proc. of Int. Conf. Comp. Vision*, pp. 3551 – 3558, 2013.

[116] F. Perronnin and C. R. Dance, "Fisher Kernels on Visual Vocabularies for Image Categorization," in *IEEE Comp. Society Conf. on Comp. Vision and Pattern Recognition*, pp. 1–8, 2007.

[117] Y. Jiang, J. Liu, A. Roshan Zamir, I. Laptev, M. Piccardi, M. Shah, and R. Sukthankar, "THUMOS Challenge: Action Recognition with a Large Number of Classes." http://crcv.ucf.edu/ICCV13-Action-Workshop/, 2013. Accessed: 2016-07-01.

[118] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition," in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 2046–2053, June 2010.

[119] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[120] H. Wang and C. Schmid, "LEAR-INRIA submission for the Thumos workshop." http://crcv.ucf.edu/THUMOS14/, 2014. Accessed: 2016-07-01.

[121] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to Document Recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[122] H. Lee, P. T. Pham, Y. Largman, and A. Y. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances of Neural Information Processing Systems*, pp. 1096–1104, 2009.

[123] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, pp. 82–97, 2012.

[124] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," pp. 1106–1114, 2012.

[125] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *In Science*, pp. 504–507, July 2006.

[126] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for Deep Belief Nets," *Neural Computing*, pp. 1527–1554, July 2006.

[127] R. R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann Machines," in *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*, pp. 448–455, 2009.

[128] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 1–9, IEEE, 2015.

[129] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. of the IEEE Conf. on Comp. Vision and Pattern Recognition*, pp. 248–255, June 2009.

[130] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *Computing Research Repository*, vol. abs/1311.2901, 2013.

[131] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning." `http://goodfeli.github.io/dlbook/`, 2016. Book in preparation for MIT Press, Accessed:2016-07-01.

[132] A. Vedaldi and A. Zisserman, "VGG Practical notes on Convolutional Neural Networks." `http://www.robots.ox.ac.uk/~vgg/practicals/cnn/`, 2015. Accessed:2016-07-01.

[133] E. A. Mosabbeb, R. Cabral, F. De la Torre, and M. Fathy, "Multi-label Discriminative Weakly-Supervised Human Activity Recognition and Localization," in *Proc. of Asian Conf. on Comp. Vision*, pp. 241–258, 2014.

[134] K. Simonyan and A. Zisserman, "Two-Stream Convolutional Networks for Action Recognition in Videos," in *Advances in Neural Information Processing Systems*, pp. 568–576, 2014.

[135] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis," in *Proc. of the Int. Conf. on Document Analysis and Recognition*, pp. 958–963, 2003.

[136] P. Sun, "Matlab toolbox for 3D Convolution and Pooling operation - MexConv3D." `https://github.com/pengsun/MexConv3D`, 2015. Accessed: 2016-07-01.

[137] A. Vedaldi and K. Lenc, "MatConvNet - Convolutional Neural Networks for MATLAB," in *Proc. of the ACM Int. Conf. on Multimedia*, 2015. Accessed: 2016-07-01.

[138] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Computing Research Repository*, vol. abs/1409.1556, 2014.